



Red Hat

Ansible Automation Platform

Simplifying Automation with Red Hat Ansible Automation Platform

An introduction to automating everything with Ansible



Red Hat
Training

Housekeeping

- Timing
- Breaks
- Takeaways

What you will learn

- Introduction to Ansible Automation
- How it works
- Understanding modules, tasks & playbooks
- How to execute Ansible commands
- Using variables & templates
- Tower - where it fits in
- Basic usage of Tower
- Learn major Tower features: RBAC, workflows and so on



Red Hat
Ansible Automation
Platform

Introduction

Topics Covered:

- What is the Ansible Automation Platform?
- What can it do?



Automation happens when one person meets a
problem they never want to solve again

Teams are automating...



Lines Of Business



Network



Security



Operations

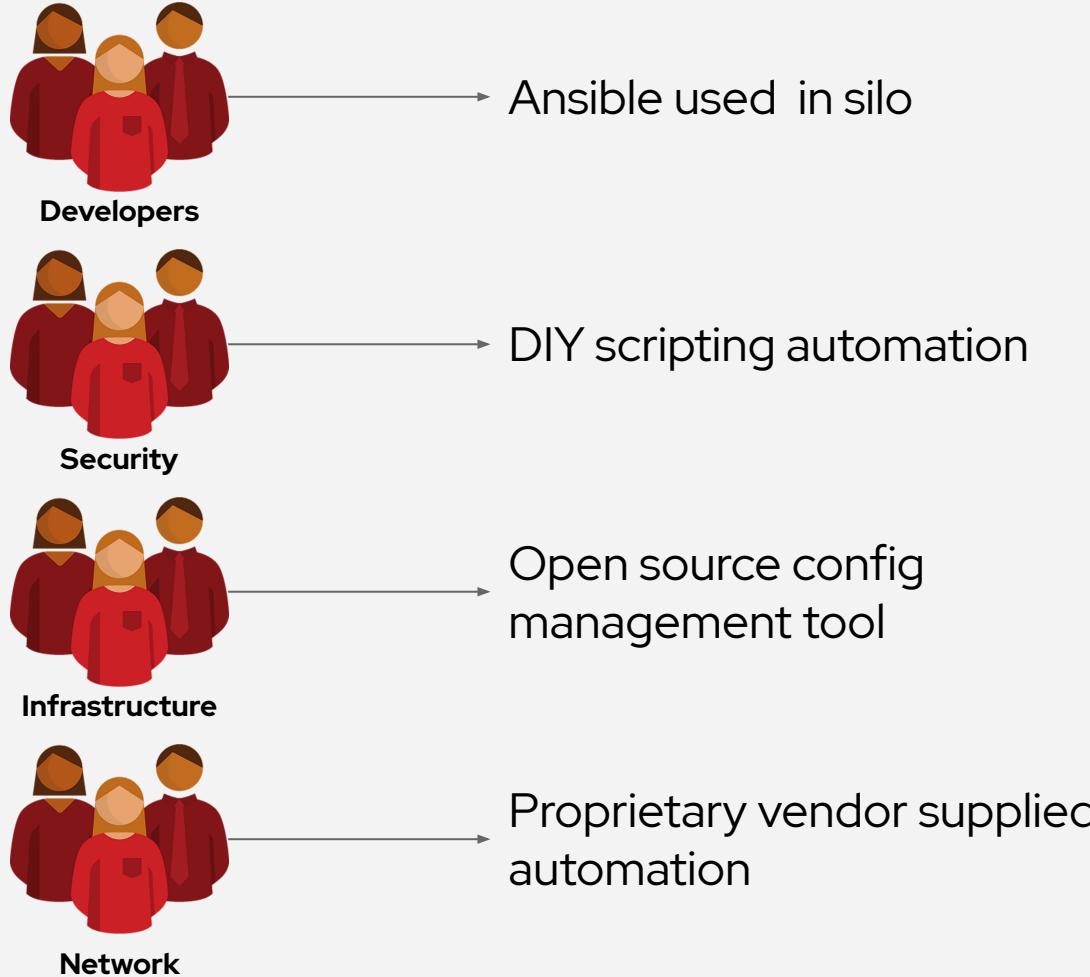


Developers



Infrastructure

Ad-hoc Automation is happening in silos



Is organic
automation enough?

Why Ansible?



Simple

Human readable automation

No special coding skills needed

Tasks executed in order

Usable by every team

Get productive quickly



Powerful

App deployment

Configuration management

Workflow orchestration

Network automation

Orchestrate the app lifecycle



Agentless

Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

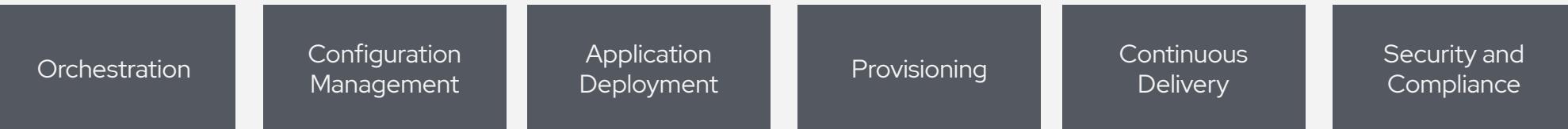
Get started immediately

More efficient & more secure

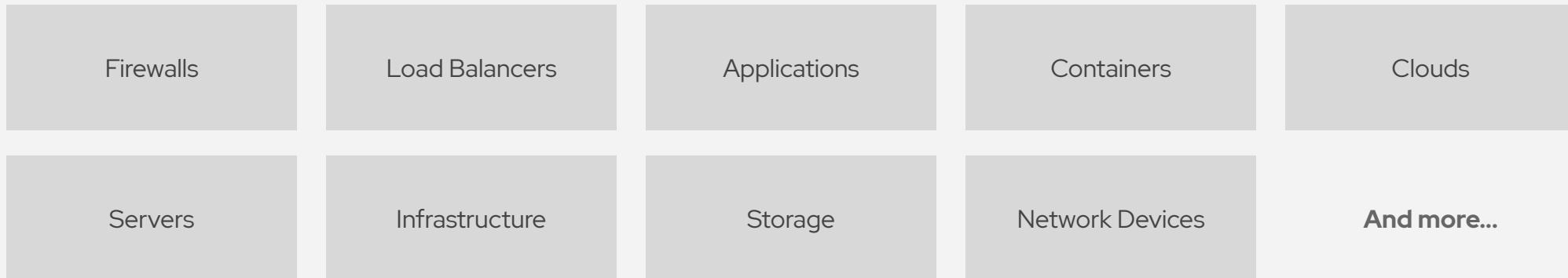
What can I do using Ansible?

Automate the deployment and management of your entire IT footprint.

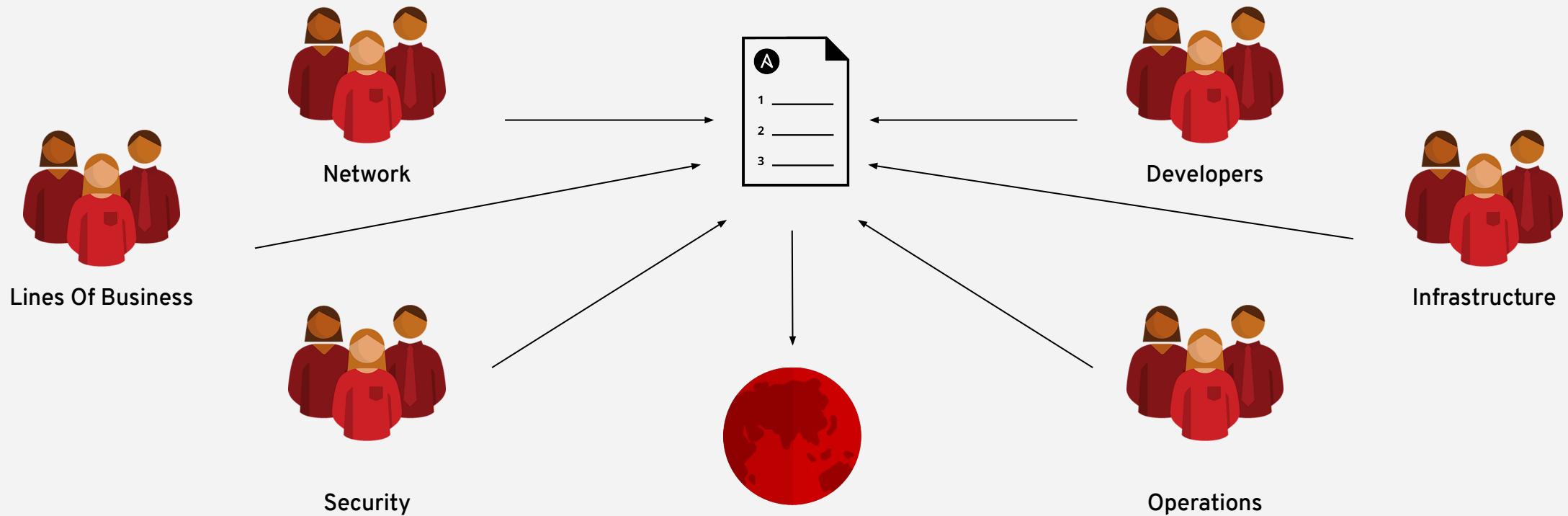
Do this...



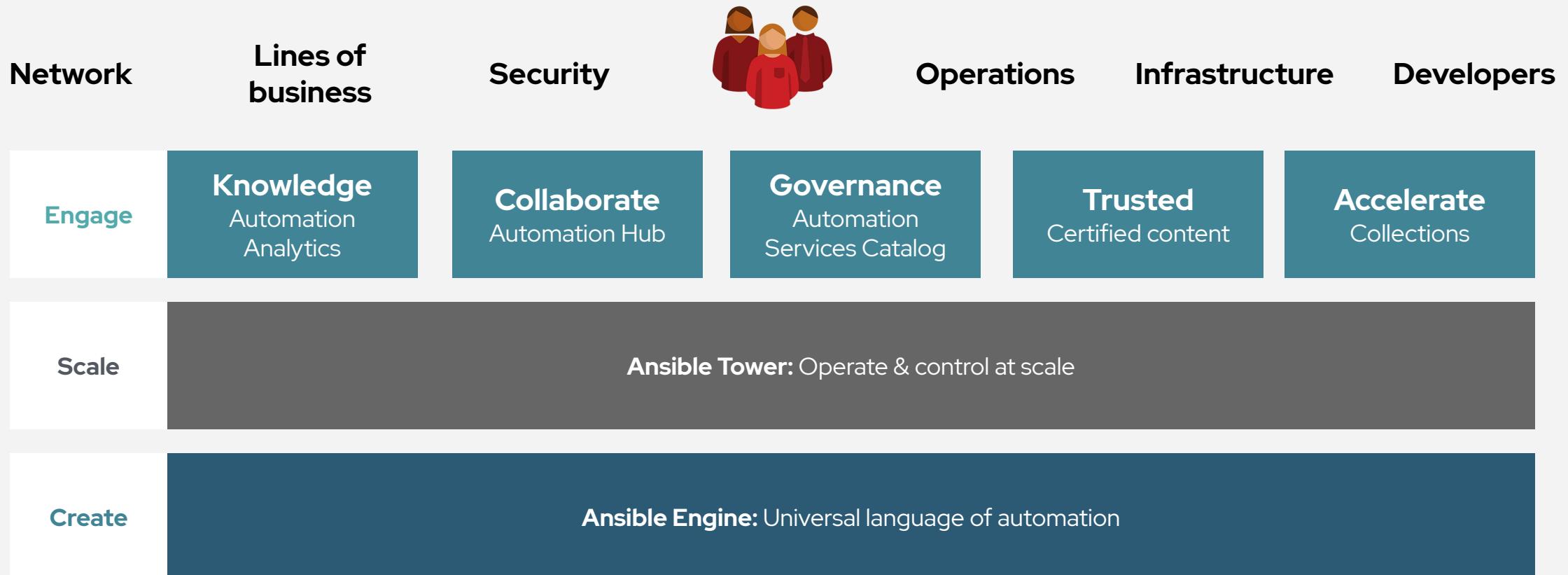
On these...



When automation crosses teams, you need an automation platform



Red Hat Ansible Automation Platform



Fueled by an open source community



Ansible automates technologies you use

Time to automate is measured in minutes

Cloud	Virt & Container	Windows	Network	Security	Monitoring
AWS	Docker	ACLs	A10	Checkpoint	Dynatrace
Azure	VMware	Files	Arista	Cisco	Datadog
Digital Ocean	RHV	Packages	Aruba	CyberArk	LogicMonitor
Google	OpenStack	IIS	Cumulus	F5	New Relic
OpenStack	OpenShift	Regedits	Bigswitch	Fortinet	Sensu
Rackspace	+more	Shares	Cisco	Juniper	+more
+more		Services	Dell	IBM	
Operating Systems	Storage	Configs	Extreme	Palo Alto	Devops
RHEL	Netapp	Users	F5	Snort	Jira
Linux	Red Hat Storage	Domains	Lenovo	+more	GitHub
Windows	Infinidat	+more	MikroTik		Vagrant
+more	+more		Juniper		Jenkins
			OpenSwitch		Slack
			+more		+more

Section 1

Ansible Engine



Red Hat
Ansible Automation
Platform

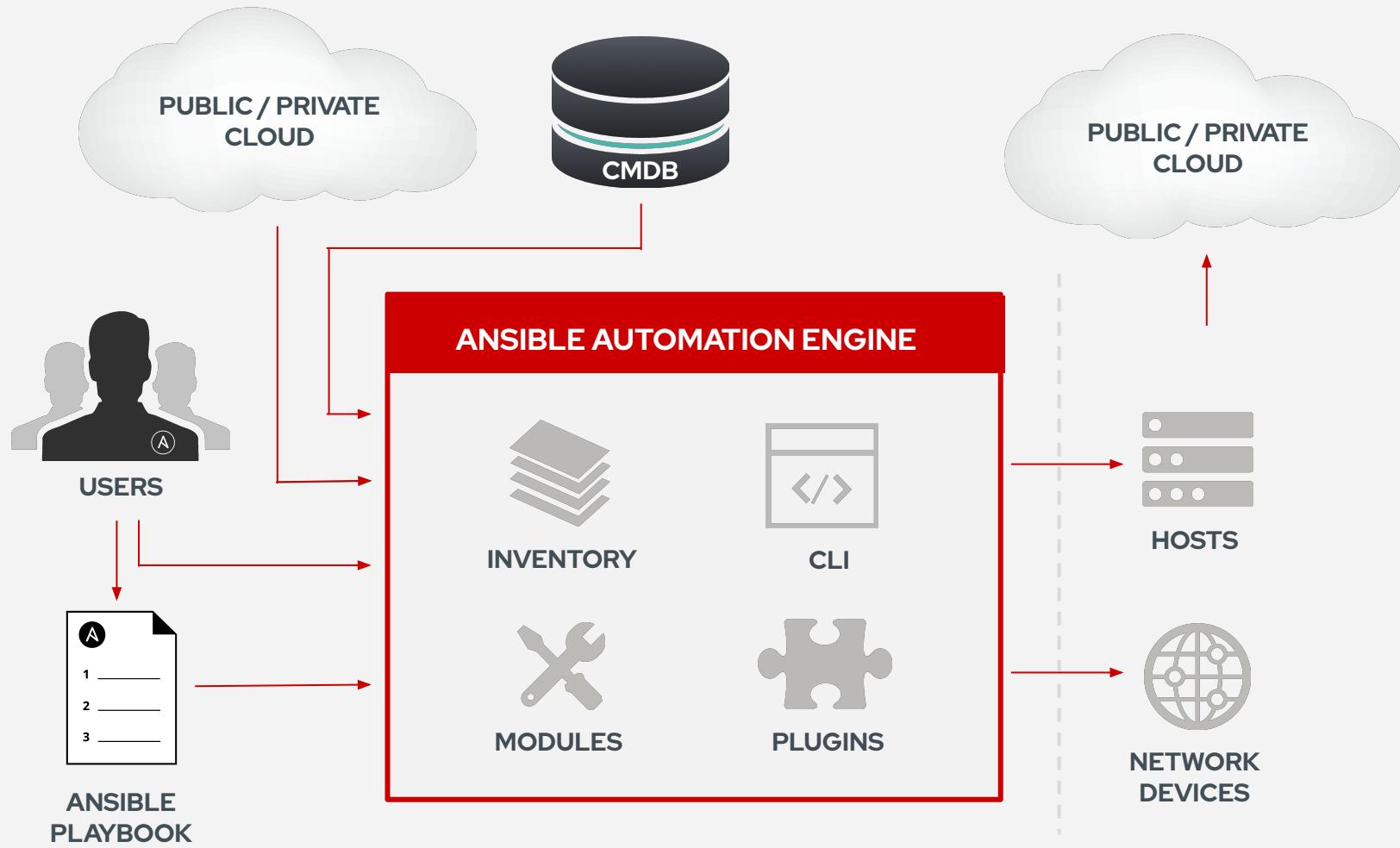


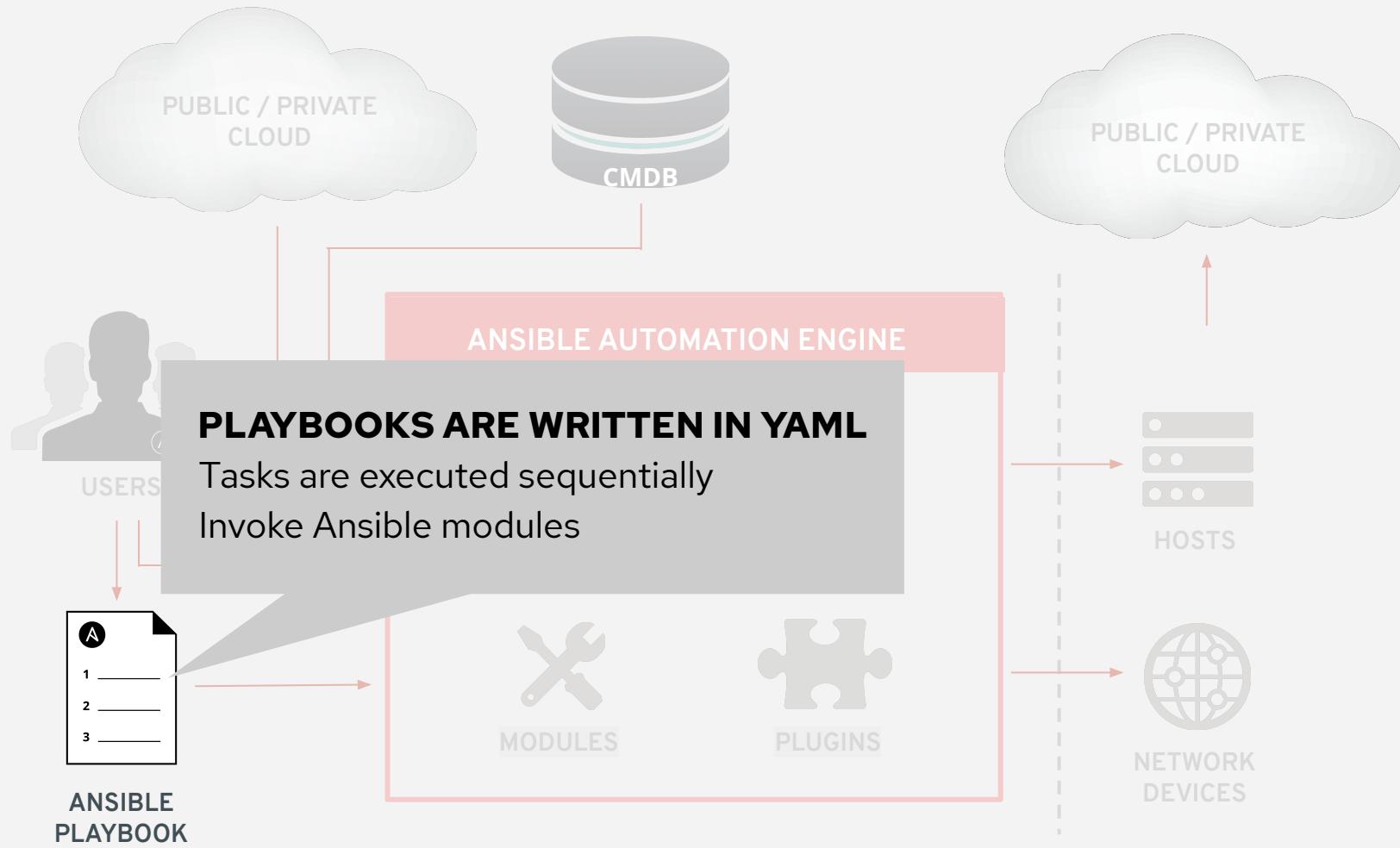
Red Hat
Ansible Automation
Platform

Section 1.1

Topics Covered:

- Understanding the Ansible Infrastructure
- Check the prerequisites



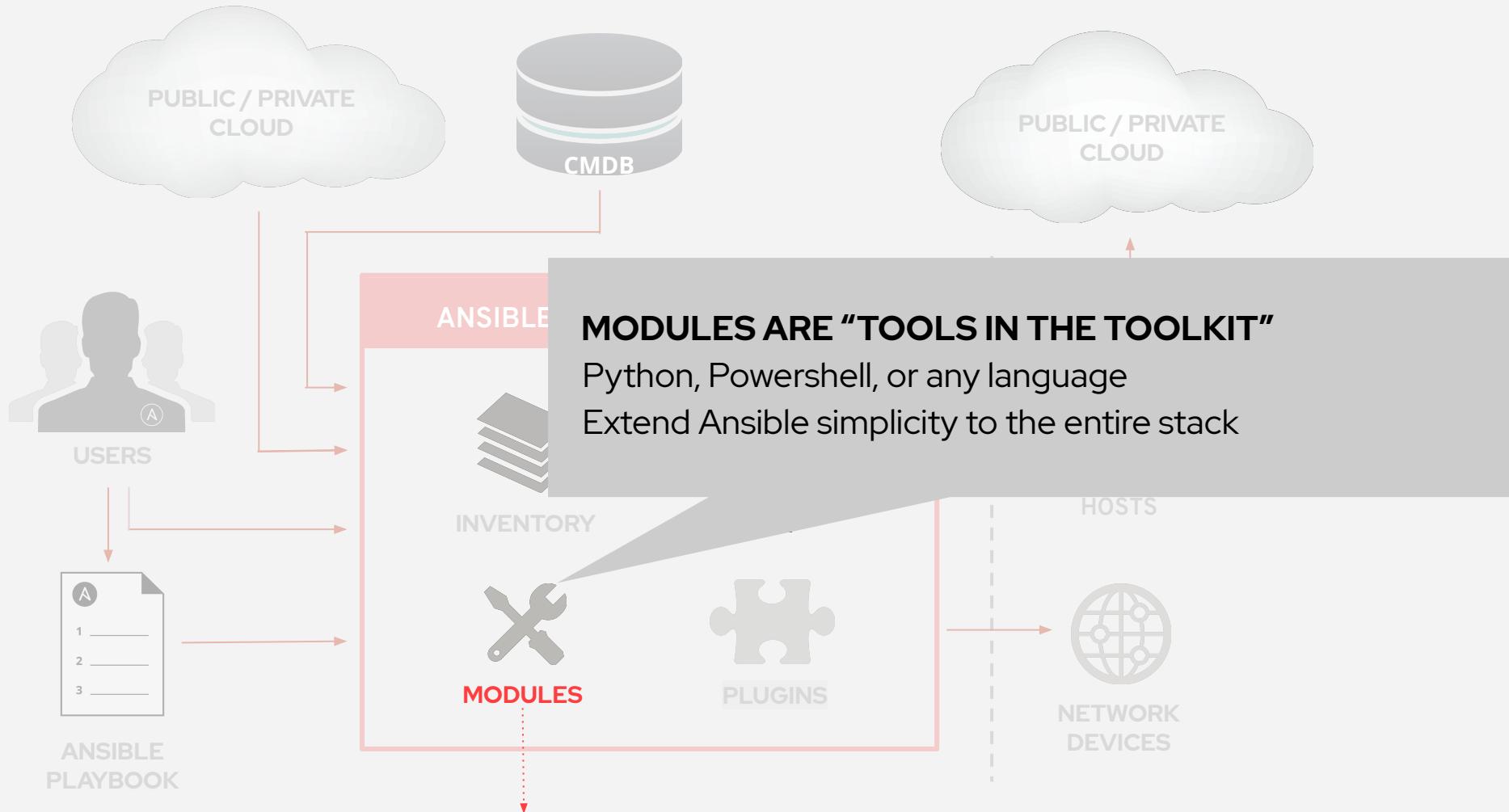


```
---
```

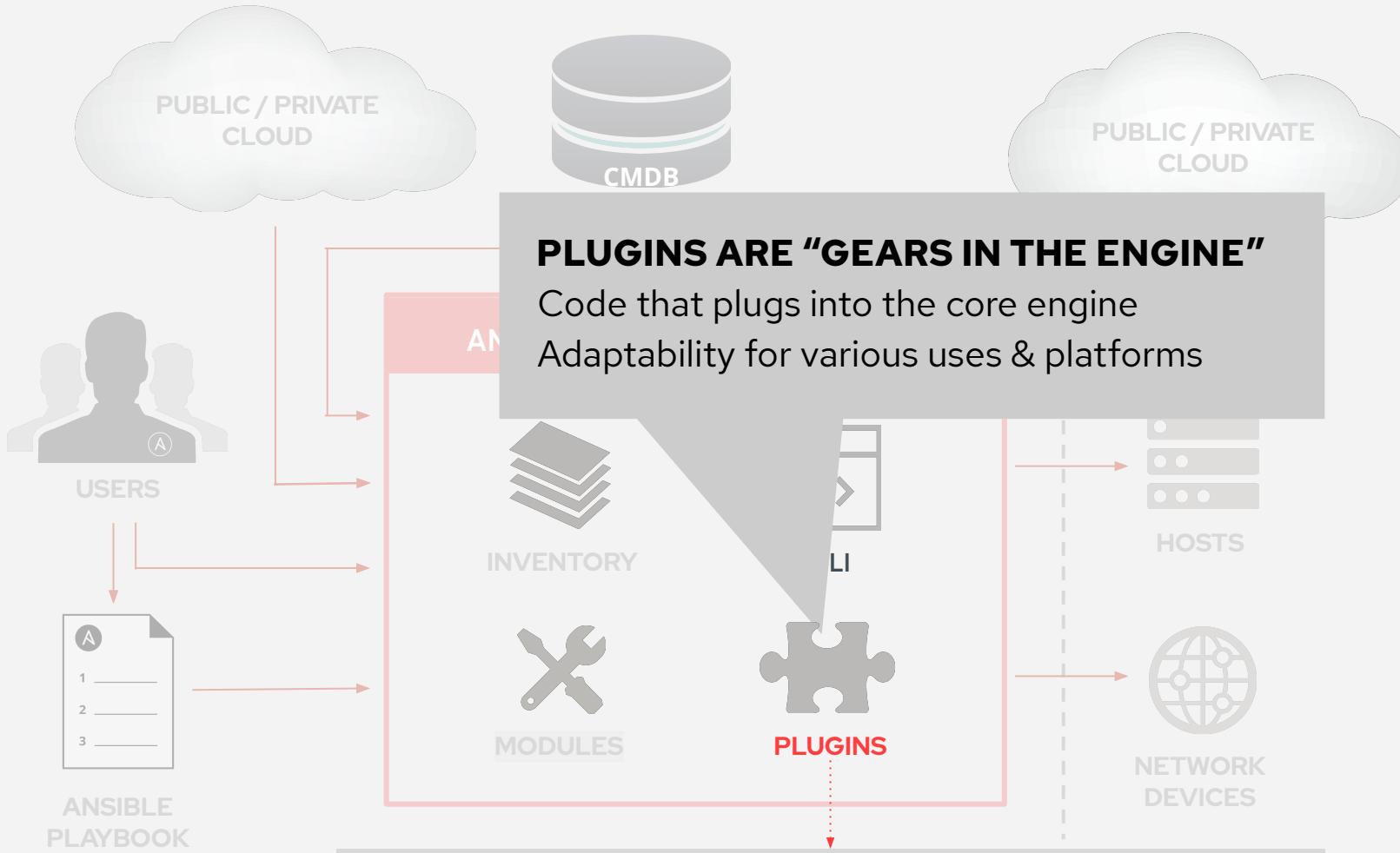
- **name: install and start apache**
hosts: web
become: yes

tasks:

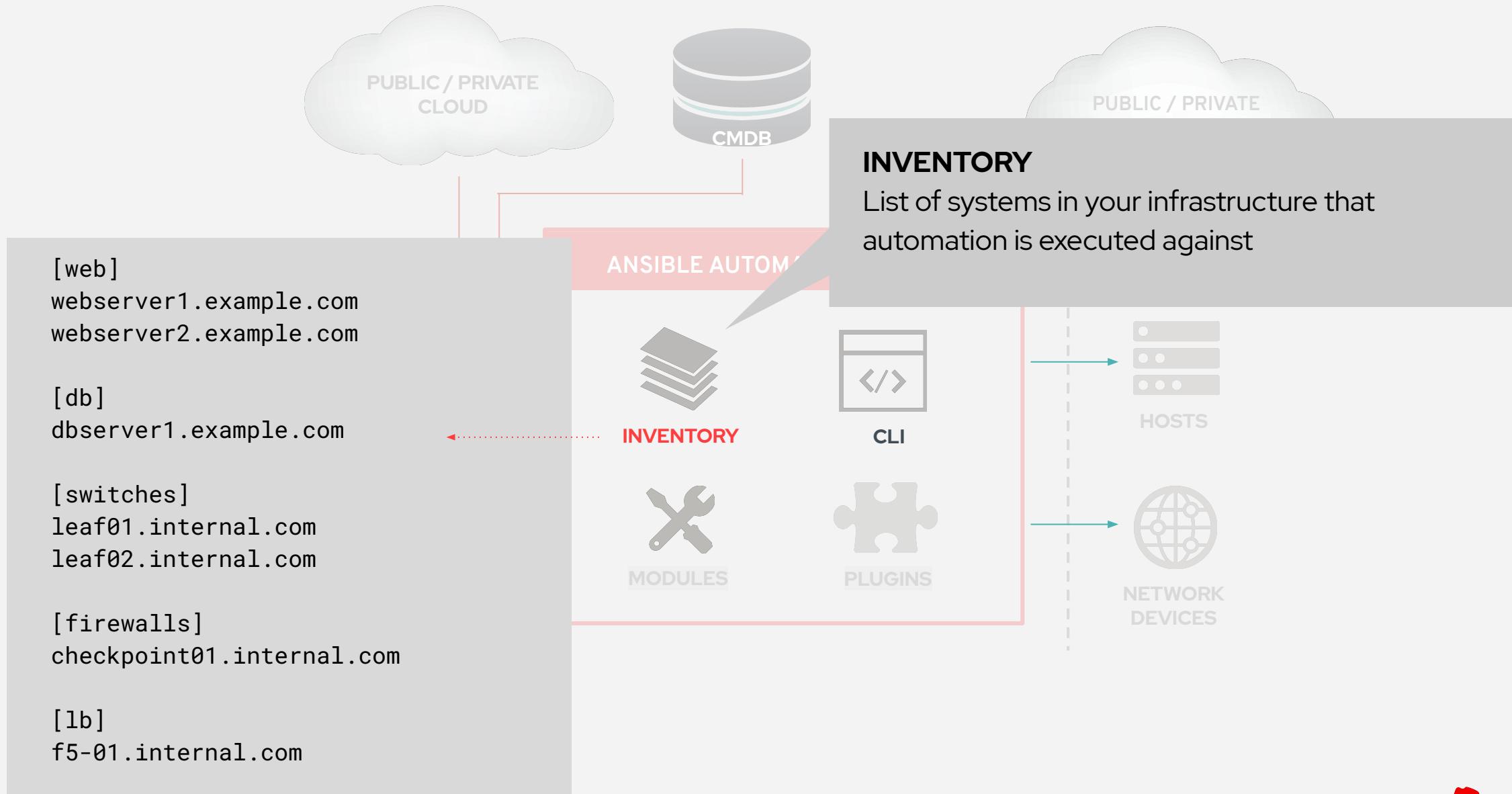
- **name: httpd package is present**
yum:
 name: httpd
 state: latest
- **name: latest index.html file is present**
template:
 src: files/index.html
 dest: /var/www/html/
- **name: httpd is started**
service:
 name: httpd
 state: started

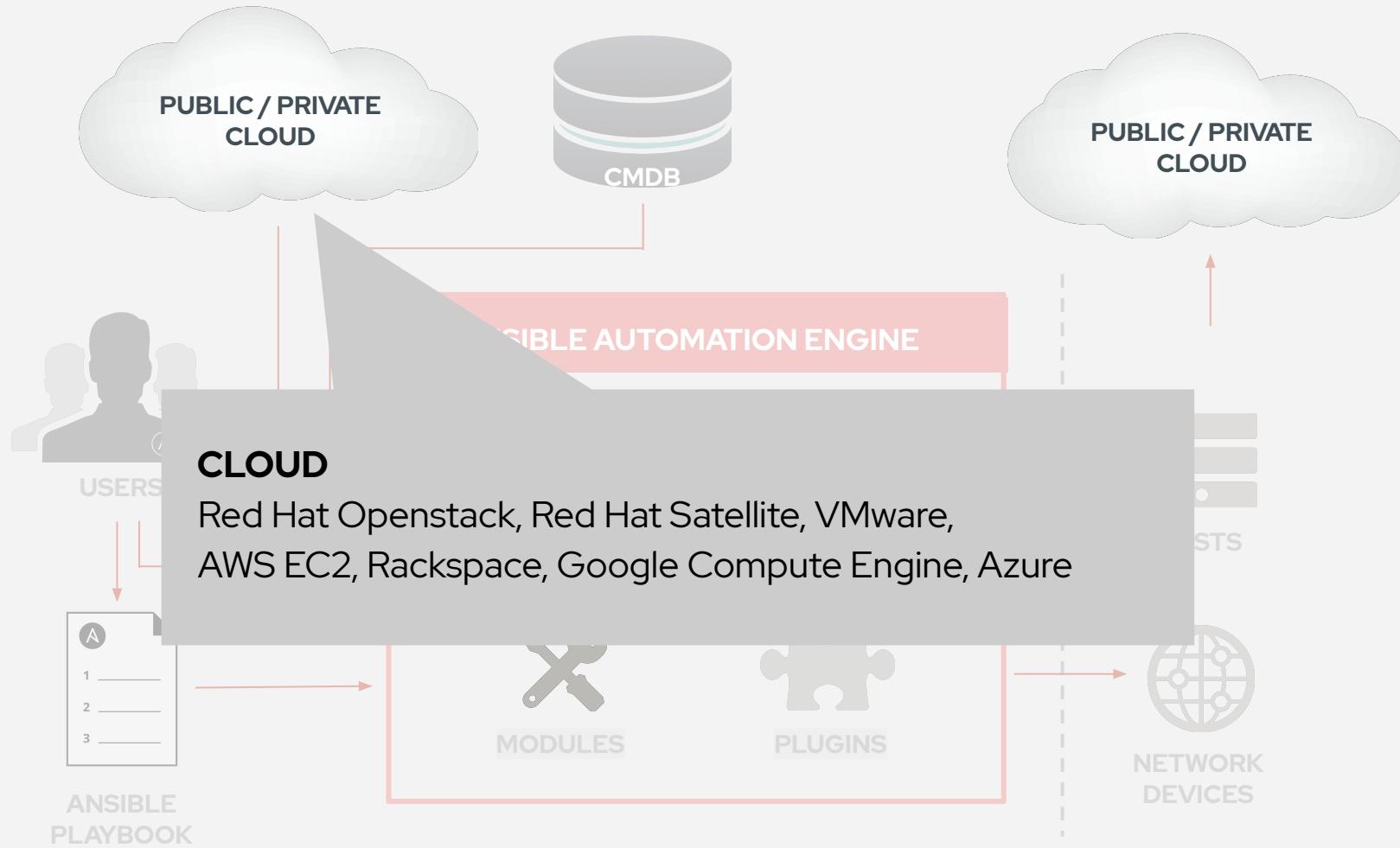


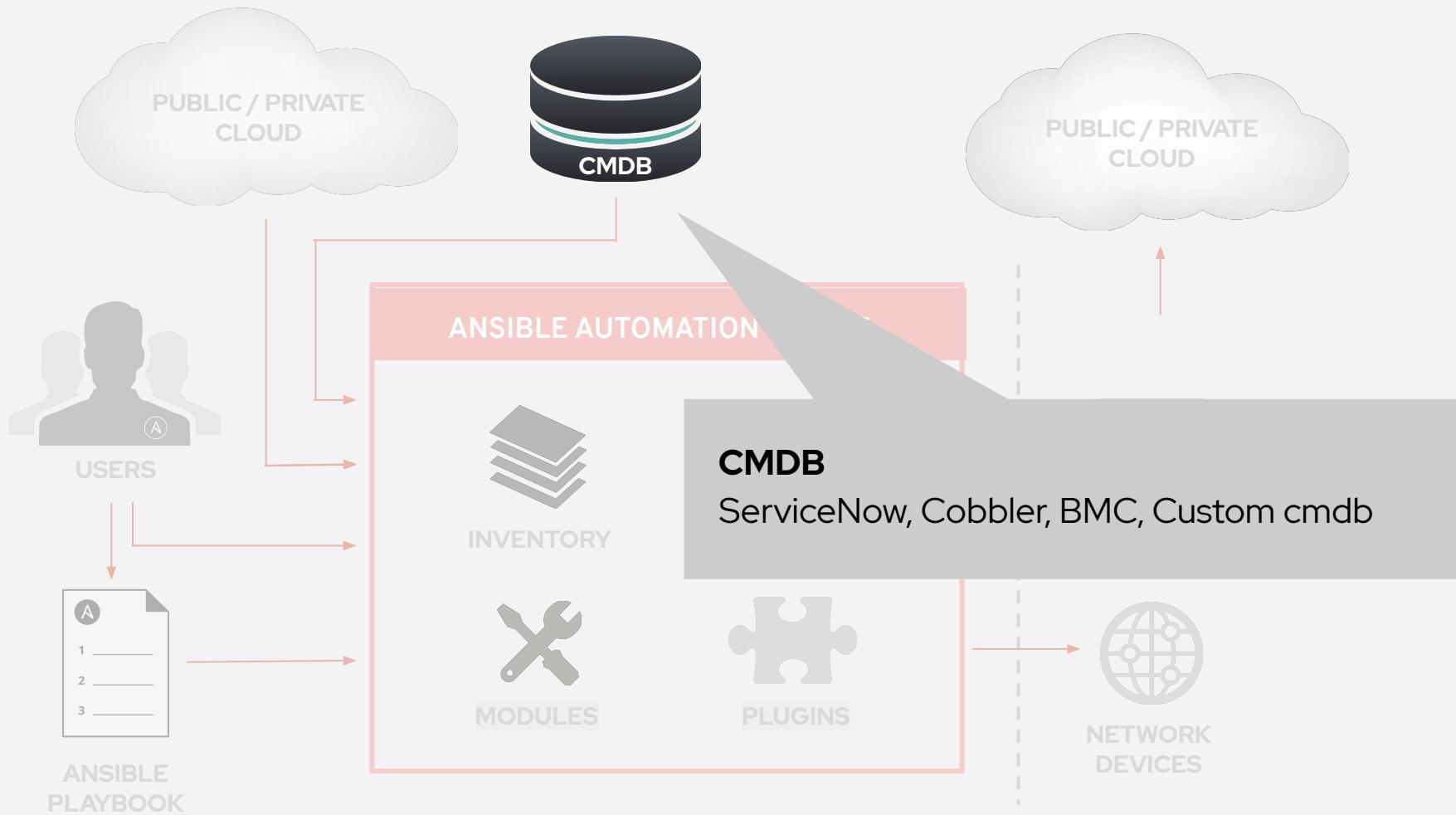
```
- name: latest index.html file is present
  template:
    src: files/index.html
    dest: /var/www/html/
```

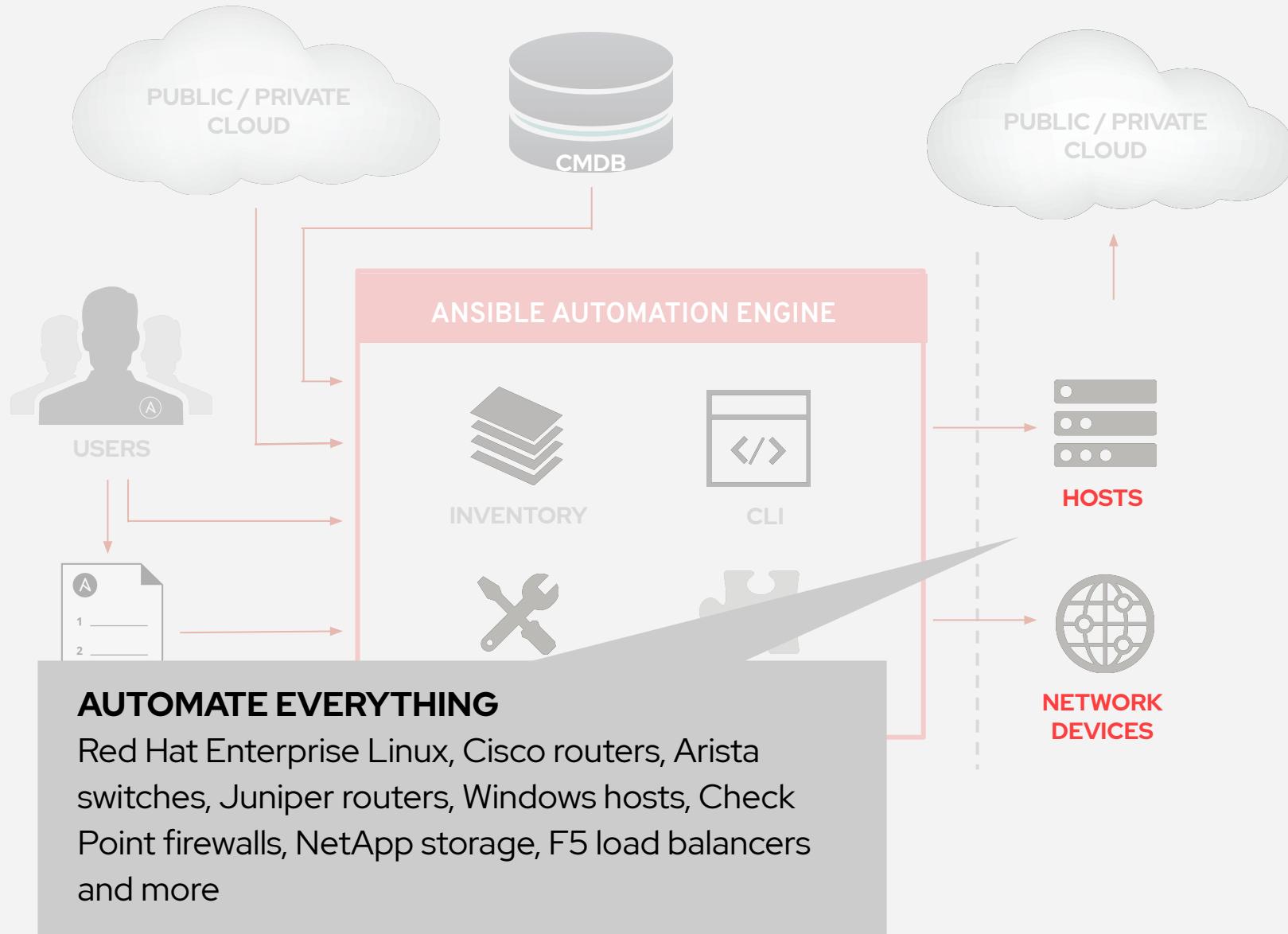


```
{ { some_variable | to_nice_yaml } }
```









LINUX AUTOMATION

150+
Linux Modules

**AUTOMATE EVERYTHING
LINUX**

**Red Hat Enterprise Linux, BSD,
Debian, Ubuntu and many more!**

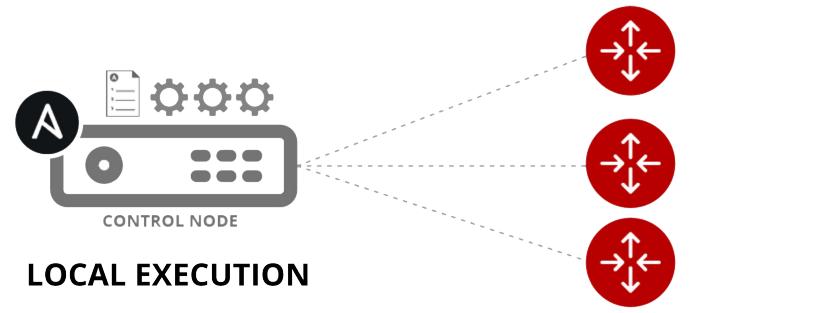
ONLY REQUIREMENTS:
Python 2 (2.6 or later)
or Python 3 (3.5 or later)

ansible.com/get-started



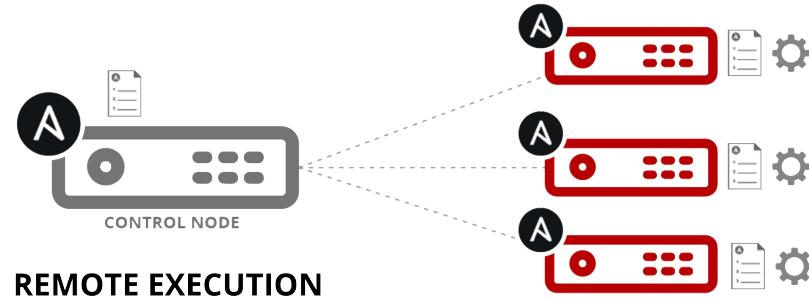
How Ansible Automation works

Module code is executed locally on the control node



**NETWORKING
DEVICES**

Module code is copied to the managed node, executed, then removed



**LINUX/WINDOWS
HOSTS**



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.1



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.2

Topics Covered:

- Ansible inventories
- Main Ansible config file
- Modules and ad-hoc commands

Inventory

- Ansible works against multiple systems in an **inventory**
- Inventory is usually file based
- Can have multiple groups
- Can have variables for each group or even host

Understanding Inventory - Basic

```
# Static inventory example:  
[myservers]  
10.42.0.2  
10.42.0.6  
10.42.0.7  
10.42.0.8  
10.42.0.100  
host.example.com
```

Understanding Inventory - Basic

[app1srv]

```
appserver01 ansible_host=10.42.0.2  
appserver02 ansible_host=10.42.0.3
```

[web]

```
node-[1:30] ansible_host=10.42.0.[31:60]
```

[web:vars]

```
apache_listen_port=8080  
apache_root_path=/var/www/mywebdocs/
```

[all:vars]

```
ansible_user=kev  
ansible_ssh_private_key_file=/home/kev/.ssh/id_rsa
```

Understanding Inventory - Variables

[app1srv]

```
appserver01 ansible host=10.42.0.2  
appserver02 ansible_host=10.42.0.3
```

[web]

```
node-[1:30] ansible_host=10.42.0.[31:60]
```

[web:vars]

```
apache_listen_port=8080  
apache_root_path=/var/www/mywebdocs/
```

[all:vars]

```
ansible_user=ender  
ansible_ssh_private_key_file=/home/ender/.ssh/id_rsa
```

Understanding Inventory - Groups

[nashville]

bnaapp01

bnaapp02

[atlanta]

atlapp03

atlapp04

[south:children]

atlanta

nashville

hsvapp05

Configuration File

- Basic configuration for Ansible
- Can be in multiple locations, with different precedence
- Here: `.ansible.cfg` in the home directory
- Configures where to find the inventory

The Ansible Configuration

Configuration files will be searched for in the following order:

- **ANSIBLE_CONFIG** (environment variable if set)
- **ansible.cfg** (in the current directory)
- **~/.ansible.cfg** (in the home directory)
- **/etc/ansible/ansible.cfg** (installed as Ansible default)

First Ad-Hoc Command: ping

- Single Ansible command to perform a task quickly directly on command line
- Most basic operation that can be performed
- Here: an example Ansible ping - not to be confused with ICMP

```
$ ansible all -m ping
```

Ad-Hoc Commands ping

```
# Check connections (submarine ping, not ICMP)
[user@ansible] $ ansible all -m ping
```

```
web1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python":
"/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

The Ansible Command

Some basics to keep you from getting stuck

--help (Display some basic and extensive options)

```
[user@ansible ~]$ ansible --help
Usage: ansible <host-pattern> [options]
```

Define and run a single task 'playbook' against a set of hosts

Options:

```
-a MODULE_ARGS, --args=MODULE_ARGS
                      module arguments
--ask-vault-pass      ask for vault password
-B SECONDS, --background=SECONDS
<<<snippet, output removed for brevity>>>
```

Ad-Hoc Commands

Here are some common options you might use:

-m MODULE_NAME , --module-name=MODULE_NAME

Module name to execute the ad-hoc command

-a MODULE_ARGS , --args=MODULE_ARGS

Module arguments for the ad-hoc command

-b , --become

Run ad-hoc command with elevated rights such as sudo, the default method

-e EXTRA_VARS , --extra-vars=EXTRA_VARS

Set additional variables as key=value or YAML/JSON

Ad-Hoc Commands

Here are some common options you might use:

```
# Check connections to all (submarine ping, not ICMP)  
[user@ansible] $ ansible all -m ping
```

```
# Run a command on all the hosts in the web group  
[user@ansible] $ ansible web -m command -a "uptime"
```

```
# Collect and display known facts for server "webl"  
[user@ansible] $ ansible webl -m setup
```



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.2



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.3

Topics Covered:

- Playbooks basics
- Running a playbook

An Ansible Playbook

A play

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```

An Ansible Playbook

A task

```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```

An Ansible Playbook

module



```
---
- name: install and start apache
  hosts: web
  become: yes

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      template:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
        state: started
```

Running an Ansible Playbook:

The most important colors of Ansible

A task executed as expected, no change was made.

A task executed as expected, making a change

A task failed to execute successfully

Running an Ansible Playbook

```
[user@ansible] $ ansible-playbook apache.yml

PLAY [webservers] ****
TASK [Gathering Facts] ****
ok: [web2]
ok: [web1]
ok: [web3]

TASK [Ensure httpd package is present] ****
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Ensure latest index.html file is present] ****
changed: [web2]
changed: [web1]
changed: [web3]

TASK [Restart httpd] ****
changed: [web2]
changed: [web1]
changed: [web3]

PLAY RECAP ****
web2          : ok=1    changed=3  unreachable=0   failed=0
web1          : ok=1    changed=3  unreachable=0   failed=0
web3          : ok=1    changed=3  unreachable=0   failed=0
```



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.3



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.4

Topics Covered:

- Working with variables
- What are facts?

An Ansible Playbook Variable Example

```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

An Ansible Playbook Variable Example

```
---
- name: variable playbook test
  hosts: localhost

  vars:
    var_one: awesome
    var_two: ansible is
    var_three: "{{ var_two }} {{ var_one }}"

  tasks:
    - name: print out var_three
      debug:
        msg: "{{var_three}}"
```

ansible is awesome



Facts

- Just like variables, really...
- ...but: coming from the host itself!
- Check them out with the setup module

```
"ansible_facts": {  
    "ansible_default_ipv4": {  
        "address": "10.41.17.37",  
        "macaddress": "00:69:08:3b:a9:16",  
        "interface": "eth0",  
    ...  
}
```



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.4



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.5

Topics Covered:

- Conditionals
- Handlers
- Loops

Conditionals via VARS

```
vars:  
  my_mood: happy  
  
tasks:  
- name: conditional task, based on my_mood var  
  debug:  
    msg: "Come talk to me. I am {{ my_mood }}!"  
  when: my_mood == "happy"
```

Conditionals with variables

```
vars:  
  my_mood: happy  
  
tasks:  
- name: conditional task, based on my_mood var  
  debug:  
    msg: "Come talk to me. I am {{ my_mood }}!"  
  when: my_mood == "happy"
```

Alternatively

```
debug:  
  msg: "Feel free to interact. I am {{ my_mood }}"  
when: my_mood != "grumpy"
```

Conditionals with facts

```
tasks:  
- name: Install apache  
  apt:  
    name: apache2  
    state: latest  
  when: ansible_distribution == 'Debian' or ansible_distribution == 'Ubuntu'  
  
- name: Install httpd  
  yum:  
    name: httpd  
    state: latest  
  when: ansible_distribution == 'RedHat'
```

Using the previous task state

This is NOT a handler task, but has similar function

- **name: Ensure httpd package is present**
yum:
 name: httpd
 state: latest
register: http_results
- **name: Restart httpd**
service:
 name: httpd
 state: restart
when: httpd_results.changed

Handler Tasks

A handler task is run when a referring task result shows a change

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
handlers:  
- name: restart_httpd  
  service:  
    name: httpd  
    state: restart
```

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **either** task notifies a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ****  
ok: [web2] unchanged  
ok: [web1] unchanged  
  
TASK [Standardized index.html file] ****  
changed: [web2]  
changed: [web1] changed  
  
NOTIFIED: [restart_httpd] ***  
changed: [web2]  
changed: [web1]
```

handler runs once

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **both** of these tasks notifies of a **changed** result, the handler will be notified **ONCE**.

```
TASK [Ensure httpd package is present] ****  
changed: [web2]  
changed: [web1] changed  
  
TASK [Standardized index.html file] ****  
changed: [web2]  
changed: [web1] changed  
  
NOTIFIED: [restart_httpd] ***  
changed: [web2]  
changed: [web1]
```

handler runs once

Handler Tasks

```
tasks:  
- name: Ensure httpd package is present  
  yum:  
    name: httpd  
    state: latest  
    notify: restart_httpd  
  
- name: Standardized index.html file  
  copy:  
    content: "This is my index.html file for {{ ansible_host }}"  
    dest: /var/www/html/index.html  
    notify: restart_httpd
```

If **neither** task notifies a **changed** result, the handler **does not run**.

```
TASK [Ensure httpd package is present] ****  
ok: [web2]  
ok: [web1]  unchanged  
  
TASK [Standardized index.html file] ****  
ok: [web2]  
ok: [web1]  unchanged  
  
PLAY RECAP ****  
web2      : ok=2  changed=0  nreachable=0  failed=0  skipped=0  rescued=0  ignored=0  
web1      : ok=2  changed=0  nreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```



Variables & Loops

Great opportunity to use a loop

```
---
```

- **name:** Ensure users
 - hosts:** node1
 - become:** yes

- tasks:**
 - **name:** Ensure user is present
 - user:**
 - name:** dev_user
 - state:** present

 - **name:** Ensure user is present
 - user:**
 - name:** qa_user
 - state:** present

 - **name:** Ensure user is present
 - user:**
 - name:** prod_user
 - state:** present

Variables & Loops

Using loops to simplify tasks

```
---
```

- **name:** Ensure users
 - hosts:** node1
 - become:** yes

tasks:

- **name:** Ensure users are present
 - user:**
 - name:** "{{item}}"
 - state:** present

loop:

- dev_user
- qa_user
- prod_user



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.5



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.6

Topics Covered:

- Templates

Variables & Templates

Using a system fact or declared variable to write a file

```
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

Variables & Templates

Using a system fact or declared variable to write a file

```
- name: Ensure apache is installed and started
  hosts: web
  become: yes
  vars:
    http_port: 80
    http_docroot: /var/www/mysite.com

  tasks:
    - name: Verify correct config file is present
      template:
        src: templates/httpd.conf.j2
        dest: /etc/httpd/conf/httpd.conf
```

```
## Excerpt from httpd.conf.j2

# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
# Listen 80    ## original line
Listen {{ http_port }}

# DocumentRoot: The directory out of which you will serve your
# documents.
# DocumentRoot "/var/www/html"
DocumentRoot {{ http_docroot }}
```



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.6



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.7

Topics Covered:

- What are roles?
- How they look like
- Galaxy

Role structure

- Defaults: default variables with lowest precedence (e.g. port)
- Handlers: contains all handlers
- Meta: role metadata including dependencies to other roles
- Tasks: plays or tasks
Tip: It's common to include tasks in main.yml with "when" (e.g. OS == xyz)
- Templates: templates to deploy
- Tests: place for playbook tests
- Vars: variables (e.g. override port)

```
user/
  └── defaults
      └── main.yml
  └── handlers
      └── main.yml
  └── meta
      └── main.yml
  └── README.md
  └── tasks
      └── main.yml
  └── templates
  └── tests
      └── inventory
          └── test.yml
  └── vars
      └── main.yml
```

Ansible Galaxy

A wide-angle photograph of a large conference hall filled with attendees seated at tables. A stage is visible in the background, featuring a large circular logo with a stylized letter 'A' and a presentation slide. The overall atmosphere is professional and focused on the event.

The image serves as a background for the central text overlay.

**Sharing
Content**

Community

**Roles, and
more**



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.7



Red Hat



Red Hat
Ansible Automation
Platform

Section 1.8

Topics Covered:

- A bonus lab - try it on your own, and when time permits



Red Hat

Ansible Automation Platform

Demo Time

Exercise 1.8



Red Hat

Section 2

Ansible Tower



Red Hat
Ansible Automation
Platform



Red Hat
Ansible Automation
Platform

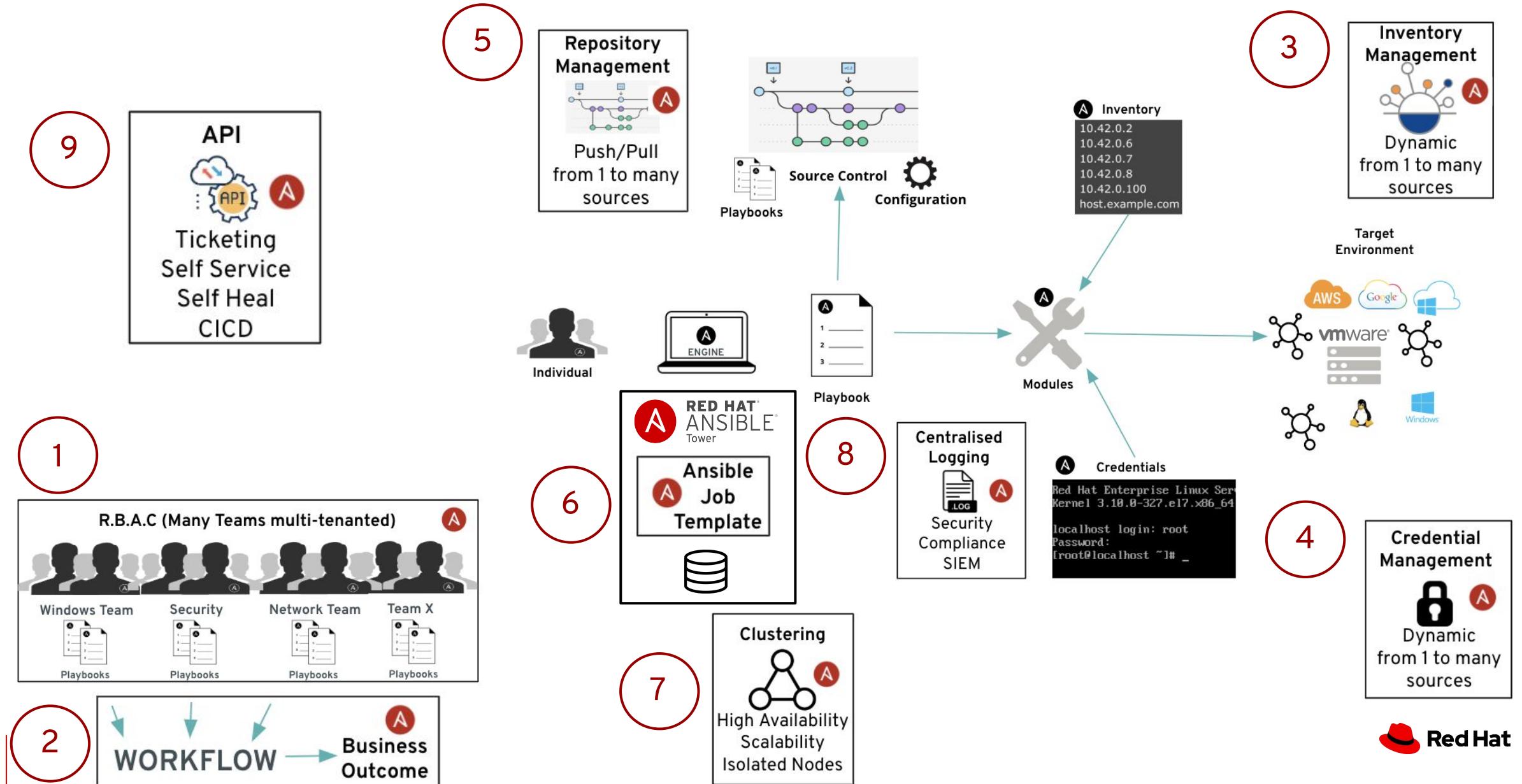
Section 2.1

Topics Covered:

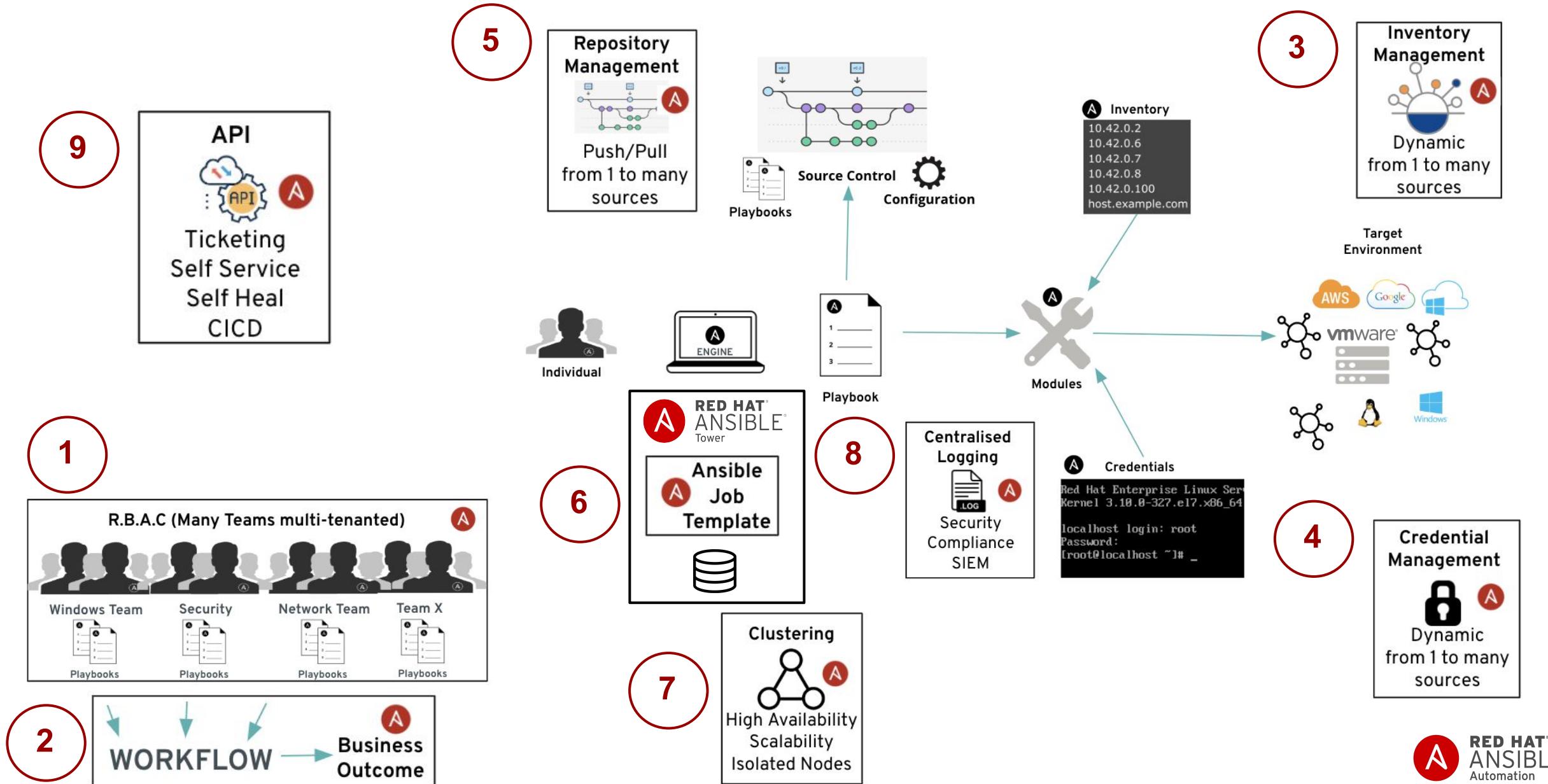
- Introduction to Tower

How Ansible Works - Ansible Tower

CONFIDENTIAL Designator



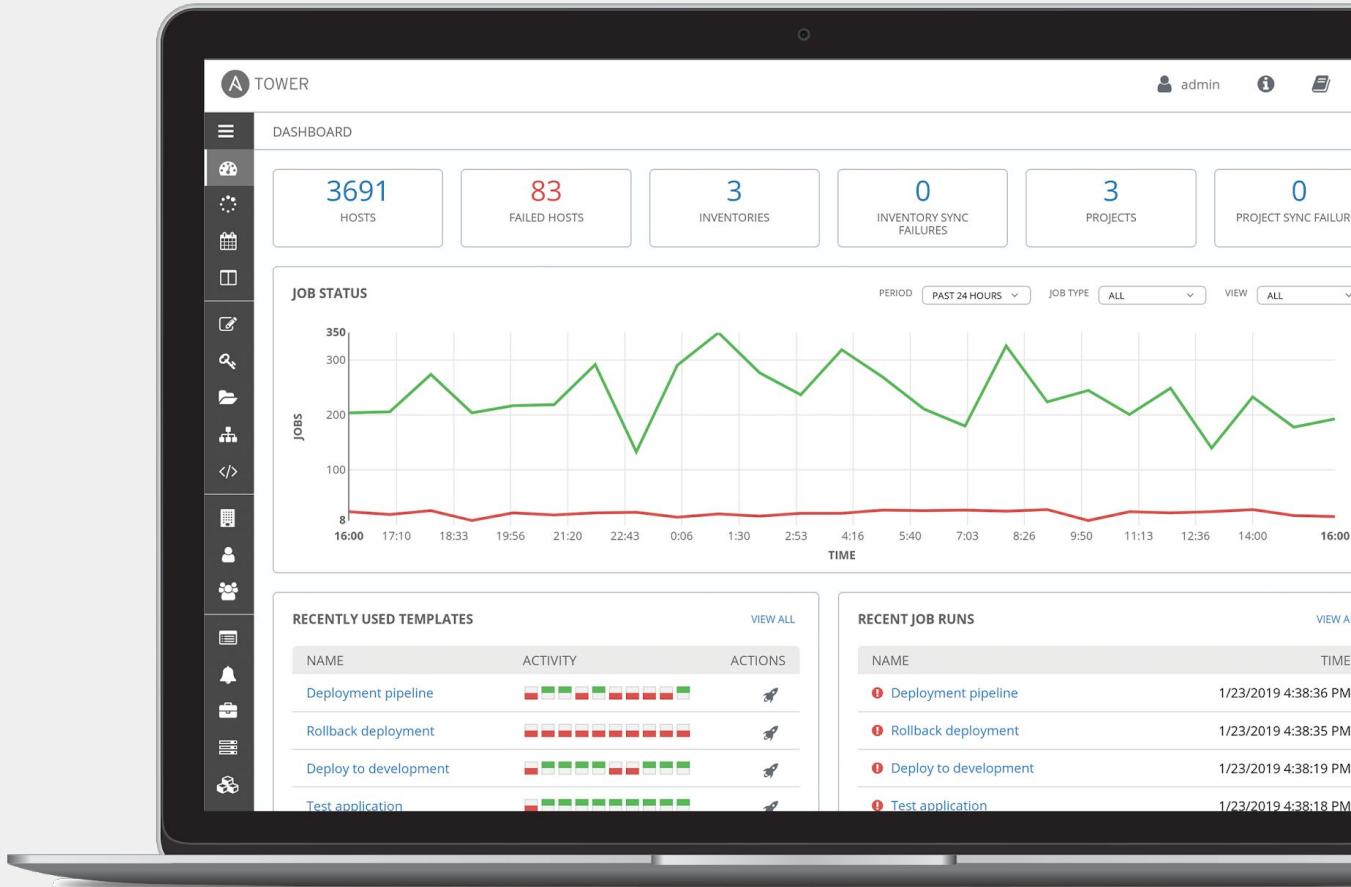
How Ansible Works - The Power of Ansible Tower



What is Ansible Tower?

Ansible Tower is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes



Red Hat Ansible Tower

Push button

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

RBAC

Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

Enterprise integrations

Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

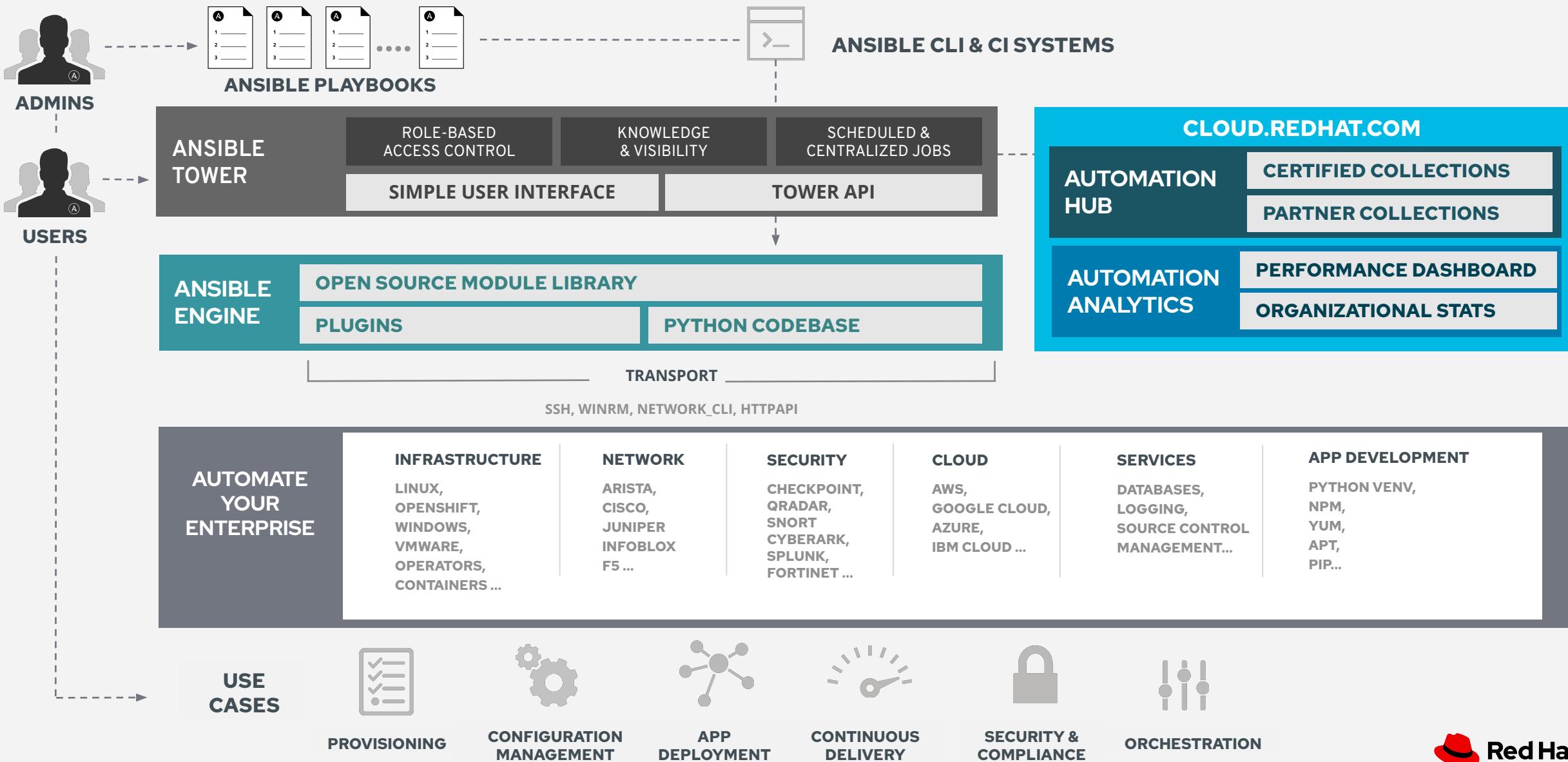
Centralized logging

All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

Workflows

Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

Ansible Automation Platform





Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.1



Red Hat



Red Hat
Ansible Automation
Platform

Section 2.2

Topics Covered:

- Inventories
- Credentials

Inventory

Inventory is a collection of hosts (nodes) with associated data and groupings that Ansible Tower can connect to and manage.

- Hosts (nodes)
- Groups
- Inventory-specific data (variables)
- Static or dynamic sources

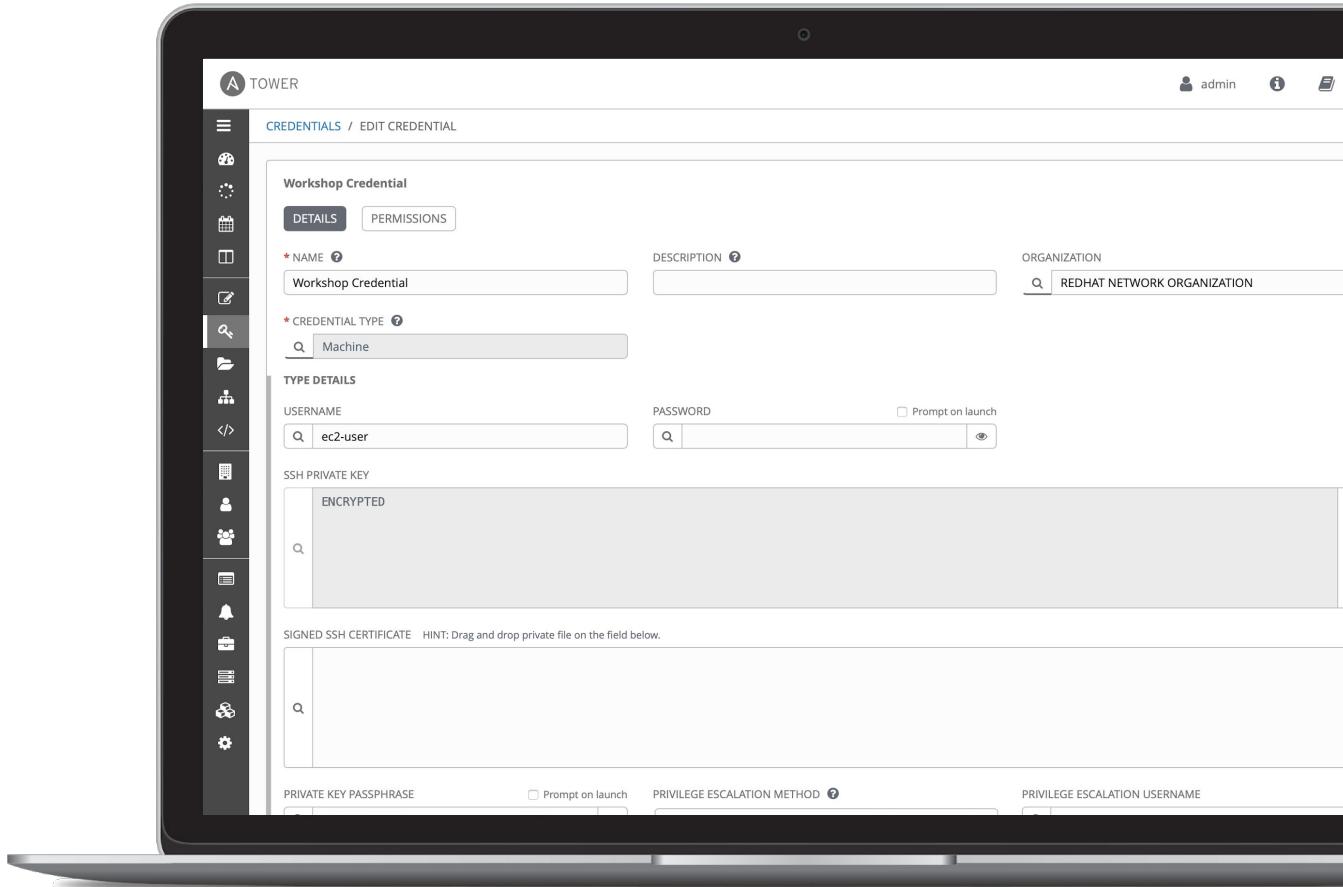
The screenshot shows the Ansible Tower web interface. The top navigation bar includes 'INVENTORIES / Workshop Inventory / HOSTS'. On the left is a sidebar with various icons for managing inventories, hosts, permissions, groups, sources, and completed jobs. The main content area displays a table titled 'Workshop Inventory' under the 'HOSTS' tab. The table lists five hosts: 'ON' (radio button), 'ansible' (radio button), 'rtr1' (radio button), 'rtr2' (radio button), 'rtr3' (radio button), and 'rtr4' (radio button). To the right of the host list are 'RELATED GROUPS' buttons: 'control' (blue), 'cisco' (blue), 'dc1' (blue), 'arista' (blue), 'dc2' (blue), 'dc1' (blue), 'juniper' (blue), 'arista' (blue), and 'dc2' (blue). Below the host table is another search bar and a table with columns for 'INVENTORIES' and 'HOSTS', showing results for 'NAME', 'TYPE', and 'ORGANIZATION'.

Credentials

Credentials are utilized by Ansible Tower for authentication with various external resources:

- Connecting to remote machines to run jobs
- Syncing with inventory sources
- Importing project content from version control systems
- Connecting to and managing network devices

Centralized management of various credentials allows end users to leverage a secret without ever exposing that secret to them.





Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.2



Red Hat



Red Hat
Ansible Automation
Platform

Section 2.3

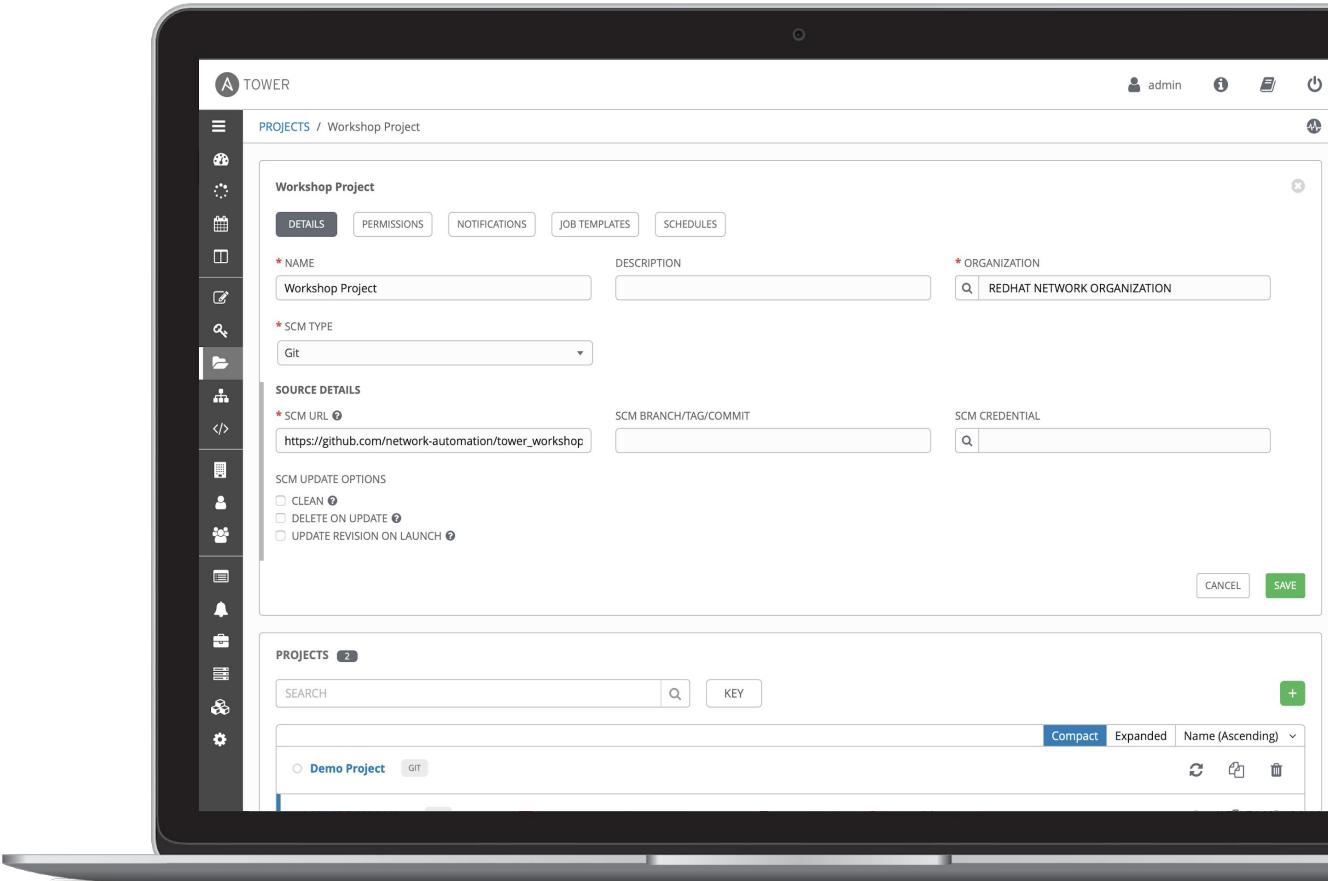
Topics Covered:

- Projects
- Job Templates

Project

A project is a logical collection of Ansible Playbooks, represented in Ansible Tower.

You can manage Ansible Playbooks and playbook directories by placing them in a source code management system supported by Ansible Tower, including Git, Subversion, and Mercurial.



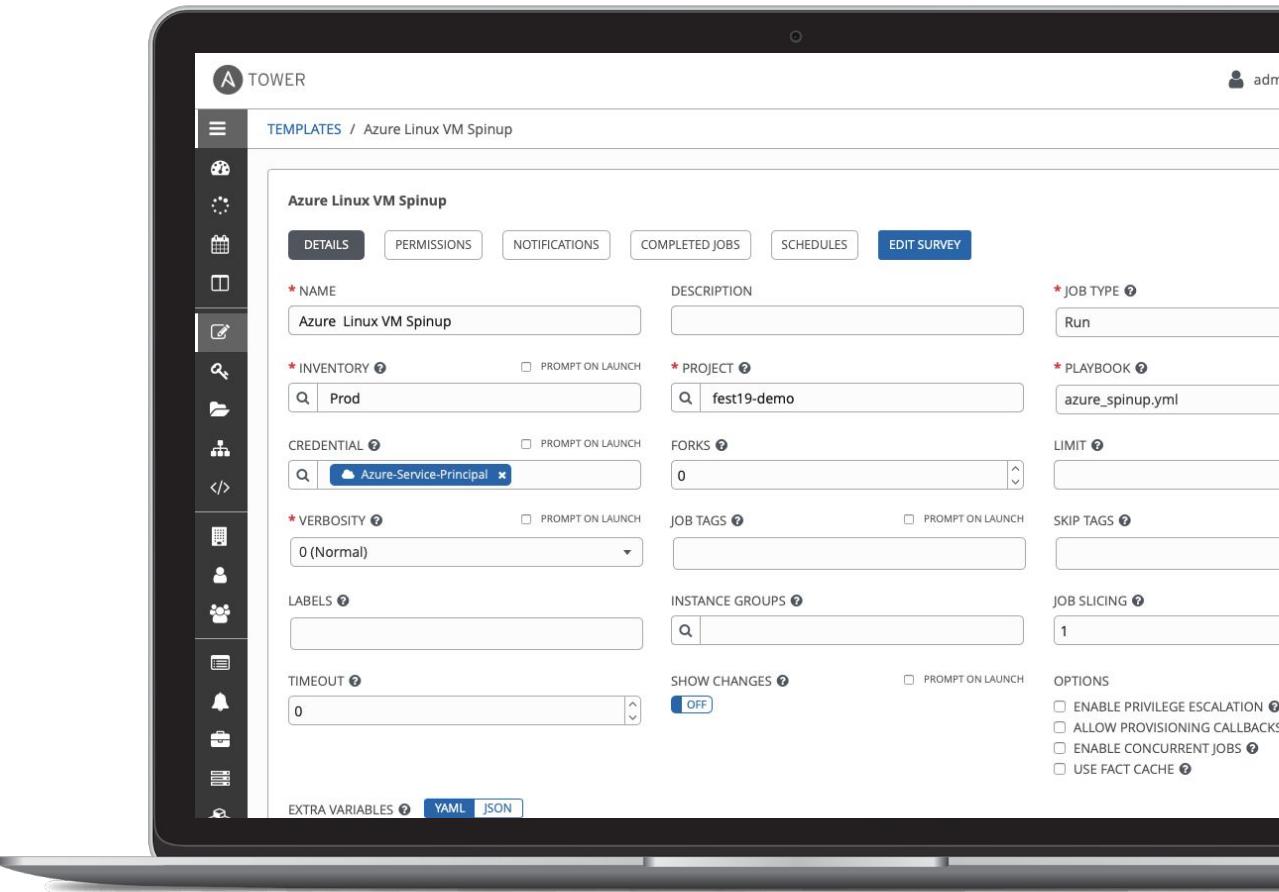
Job Templates

Everything in Ansible Tower revolves around the concept of a **Job Template**. Job Templates allow Ansible Playbooks to be controlled, delegated and scaled for an organization.

Job templates also encourage the reuse of Ansible Playbook content and collaboration between teams.

A **Job Template** requires:

- An **Inventory** to run the job against
- A **Credential** to login to devices.
- A **Project** which contains Ansible Playbooks



Expanding on Job Templates

Job Templates can be found and created by clicking the **Templates** button under the *RESOURCES* section on the left menu.



The screenshot shows the Ansible Tower web interface. The left sidebar has a dark theme with white icons and labels. The 'RESOURCES' section is expanded, showing 'Templates' as the active item, which is highlighted with a light blue background. Other items in this section include 'Credentials', 'Projects', 'Inventories', 'Inventory Scripts', 'Organizations', 'Users', and 'Teams'. The main content area is titled 'TEMPLATES' and shows a list of six job templates: 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. Each template entry includes a small thumbnail icon, the template name, a 'Job Template' label, and three action icons: a rocket (Run), a document (Edit), and a trash can (Delete). Above the list are search and key filters, and below it are sorting options ('Compact', 'Expanded', 'Name (Ascending)'). The top right corner shows the user 'admin' and various system status icons. At the bottom right, it says 'ITEMS 1 - 6'.

Executing an existing Job Template

Job Templates can be launched by clicking the **rocketship button** for the corresponding Job Template



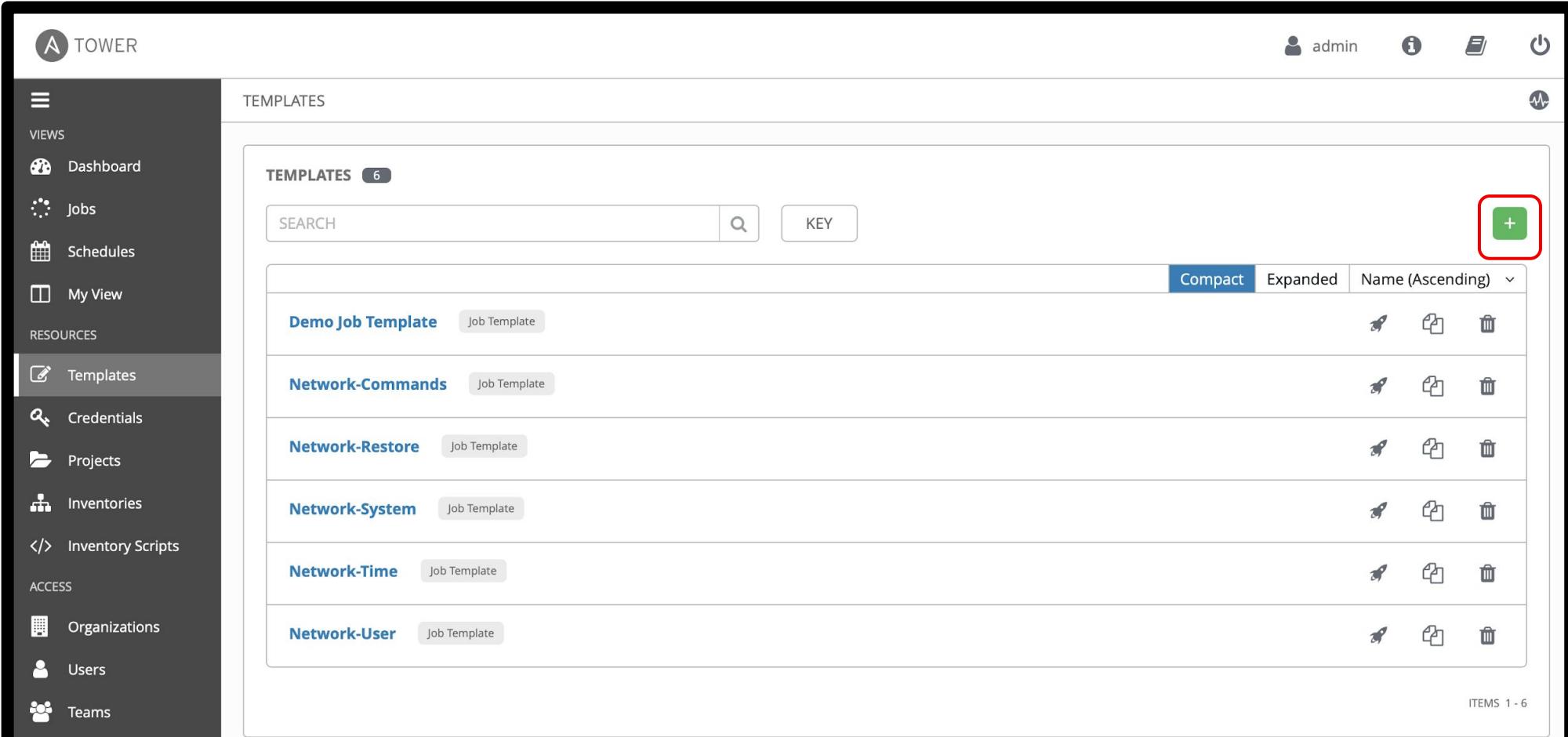
The screenshot shows the Tower interface with the 'TEMPLATES' view selected. The left sidebar includes options like 'Dashboard', 'Jobs', 'Schedules', 'My View', 'Templates' (which is currently selected), 'Credentials', 'Projects', 'Inventories', 'Inventory Scripts', 'Organizations', 'Users', and 'Teams'. The main area displays a list of six job templates:

Template Name	Type	Action Icons
Demo Job Template	Job Template	Rocketship, Copy, Delete
Network-Commands	Job Template	Rocketship, Copy, Delete
Network-Restore	Job Template	Rocketship, Copy, Delete
Network-System	Job Template	Rocketship, Copy, Delete
Network-Time	Job Template	Rocketship, Copy, Delete
Network-User	Job Template	Rocketship, Copy, Delete

A red box highlights the 'Rocketship' icon in the first row's actions column. The bottom right corner of the interface shows 'ITEMS 1 - 6'.

Creating a new Job Template (1/2)

New Job Templates can be created by clicking the **plus button**



The screenshot shows the Ansible Tower web interface. On the left is a dark sidebar with navigation links: Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration. The 'Templates' link is currently selected. The main content area is titled 'TEMPLATES' and shows a list of six existing job templates: 'Demo Job Template', 'Network-Commands', 'Network-Restore', 'Network-System', 'Network-Time', and 'Network-User'. Each template entry includes a 'Job Template' badge and three icons for edit, copy, and delete. In the top right corner of the template list area, there is a green button with a white plus sign (+). This button is highlighted with a red square, indicating it is the target for creating a new job template. The top navigation bar includes a user icon labeled 'admin', a help icon, a settings icon, and a power icon. At the bottom right of the main content area, it says 'ITEMS 1 - 6'.

Creating a new Job Template (2/2)

This **New Job Template** window is where the inventory, project and credential are assigned. The red asterisk * means the field is required.

The screenshot shows the 'New Job Template' configuration window. On the left is a sidebar with navigation links for Views, Resources, Access, and Administration. The 'Templates' link in the Resources section is highlighted. The main window has tabs for Details, Permissions, Completed Jobs, Schedules, and Add Survey. The Details tab is active. It contains fields for Name, Description, Job Type (Run), Inventory, Project, Playbook, Credential, Forks, Limit, Verbosity, Job Tags, Skip Tags, Labels, Instance Groups, Job Slicing, Timeout, Show Changes, and Options (Enable Privilege Escalation, Allow Provisioning Callbacks). Most fields have a 'Prompt on Launch' checkbox. Red asterisks (*) are placed next to the required fields: Name, Inventory, Project, Playbook, Verbosity, and Job Slicing.

NEW JOB TEMPLATE

DETAILS PERMISSIONS COMPLETED JOBS SCHEDULES ADD SURVEY

* NAME

DESCRIPTION

* JOB TYPE Run

* INVENTORY PROMPT ON LAUNCH

* PROJECT PROMPT ON LAUNCH

* PLAYBOOK Choose a playbook

CREDENTIAL PROMPT ON LAUNCH

FORKS 0

LIMIT PROMPT ON LAUNCH

* VERBOSITY 0 (Normal)

JOB TAGS PROMPT ON LAUNCH

SKIP TAGS PROMPT ON LAUNCH

LABELS

INSTANCE GROUPS PROMPT ON LAUNCH

JOB SLICING 1

TIMEOUT PROMPT ON LAUNCH

SHOW CHANGES OFF

OPTIONS

ENABLE PRIVILEGE ESCALATION

ALLOW PROVISIONING CALLBACKS



Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.3



Red Hat



Red Hat
Ansible Automation
Platform

Section 2.4

Topics Covered:

- Surveys

Surveys

Tower surveys allow you to configure how a job runs via a series of questions, making it simple to customize your jobs in a user-friendly way.

An Ansible Tower survey is a simple question-and-answer form that allows users to customize their job runs. Combine that with Tower's role-based access control, and you can build simple, easy self-service for your users.

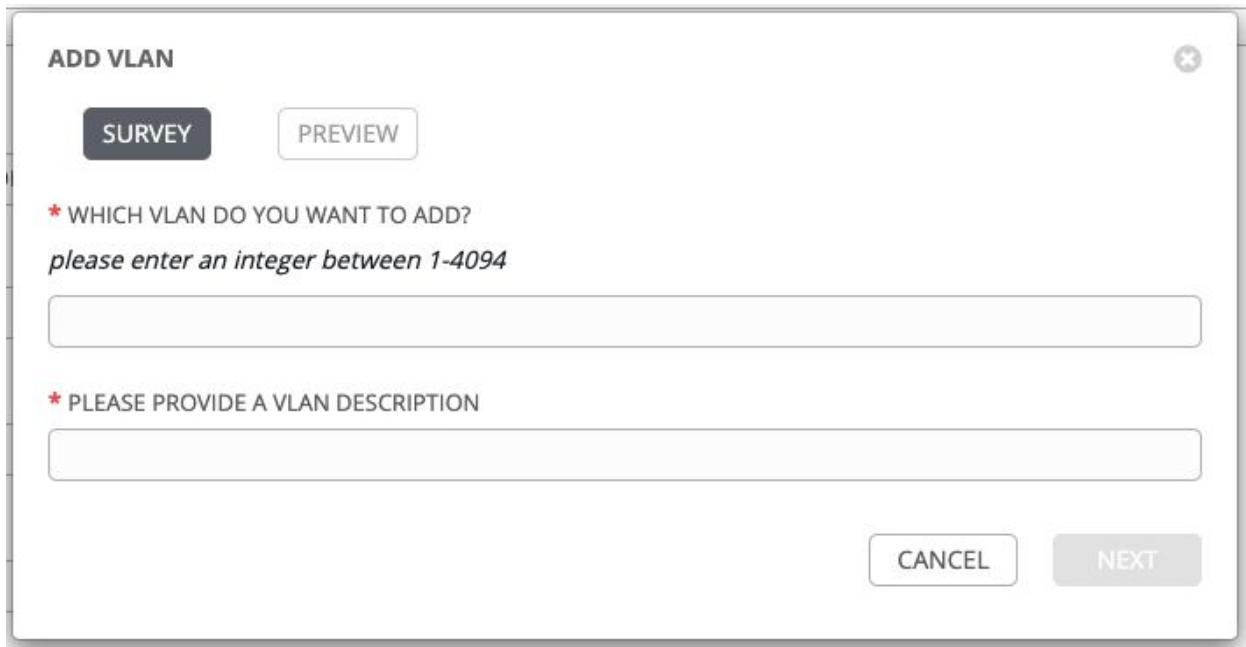
ADD VLAN

SURVEY PREVIEW

* WHICH VLAN DO YOU WANT TO ADD?
please enter an integer between 1-4094

* PLEASE PROVIDE A VLAN DESCRIPTION

CANCEL NEXT



Creating a Survey (1/2)

Once a Job Template is saved, the **Add Survey Button** will appear

ADD SURVEY

Click the button to open the Add Survey window.

The screenshot shows the Ansible Tower web interface. On the left is a dark sidebar with navigation links: Views, Dashboard, Jobs, Schedules, My View, Resources, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Access, and Organizations. The main content area has a title 'TEMPLATES / Configure Banner'. A modal window titled 'Configure Banner' is open. Inside the modal, there are several configuration sections: 'DETAILS' (selected), 'PERMISSIONS', 'NOTIFICATIONS', 'COMPLETED JOBS', 'SCHEDULES', and 'EDIT SURVEY' (which is highlighted with a red rectangle). Below these are fields for 'NAME' ('Configure Banner'), 'DESCRIPTION', 'JOB TYPE' ('Run'), 'INVENTORY' ('Workshop Inventory'), 'PROJECT' ('Workshop Project'), 'PLAYBOOK' ('network_banner.yml'), 'CREDENTIAL' ('Workshop Credential'), 'FORKS' ('0'), 'LIMIT' (empty), 'VERBOSITY' ('0 (Normal)'), 'JOB TAGS' (empty), 'SKIP TAGS' (empty), and 'LABELS' (empty). There are also 'PROMPT ON LAUNCH' checkboxes for various fields.

Creating a Survey (2/2)

The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.

The screenshot shows the 'Edit Survey Prompt' configuration window. At the top left, there's a 'CONFIGURE BANNER | SURVEY ON' button. The main area is titled 'EDIT SURVEY PROMPT' and contains the following fields:

- * PROMPT**: A text input field containing "Please enter the banner text".
- DESCRIPTION**: A text input field containing "Please type into the text field the desired banner".
- * ANSWER VARIABLE NAME**: A text input field containing "net_banner".
- * ANSWER TYPE**: A dropdown menu showing "Textarea".
- MINIMUM LENGTH**: A numeric input field set to "0".
- MAXIMUM LENGTH**: A numeric input field set to "4096".
- DEFAULT ANSWER**: An empty text input field.

At the bottom left, there's a checked checkbox labeled "REQUIRED". At the bottom right, there are "CLEAR" and "UPDATE" buttons. On the right side of the window, there's a 'PREVIEW' section with a large text input field containing "Please enter the banner text" and "Please type into the text field the desired banner". This preview field has a blue edit icon and a trash icon. Below the preview is a small "..." icon.

Creating a Survey (2/2)

The Add Survey window allows the Job Template to prompt users for one or more questions. The answers provided become variables for use in the Ansible Playbook.

The screenshot shows the 'Edit Survey Prompt' configuration window. It includes fields for 'PROMPT' (containing 'Please enter the banner text'), 'DESCRIPTION' (containing 'Please type into the text field the desired banner'), 'ANSWER VARIABLE NAME' (set to 'net_banner'), 'ANSWER TYPE' (set to 'Textarea'), 'MINIMUM LENGTH' (set to 0), 'MAXIMUM LENGTH' (set to 4096), and a 'DEFAULT ANSWER' field. A 'REQUIRED' checkbox is checked. On the right, a 'PREVIEW' panel shows the survey prompt with the same text and instructions, along with edit and delete icons.

CONFIGURE BANNER | SURVEY ON

EDIT SURVEY PROMPT

* PROMPT
Please enter the banner text

DESCRIPTION
Please type into the text field the desired banner

* ANSWER VARIABLE NAME ⓘ
net_banner

* ANSWER TYPE ⓘ
Textarea

MINIMUM LENGTH
0

MAXIMUM LENGTH
4096

DEFAULT ANSWER

REQUIRED

PREVIEW

* PLEASE ENTER THE BANNER TEXT
Please type into the text field the desired banner

⋮ edit delete

CLEAR UPDATE CANCEL SAVE

Using a Survey

When launching a job, the user will now be prompted with the Survey. The user can be required to fill out the Survey before the Job Template will execute.

The screenshot shows the TOWER interface. On the left is a dark sidebar with various navigation options: Views, Dashboard, Jobs, Schedules, My View, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Organizations, Users, and Teams. The main area is titled 'TEMPLATES' and lists four job templates: 'Network-Restore' (Job Template), 'Network-System' (Job Template), 'Network-Time' (Job Template), and 'Network-User' (Job Template). To the right of these templates is a 'CONFIGURE BANNER' dialog box. The dialog has tabs for 'SURVEY' (which is selected) and 'PREVIEW'. It contains a text field with the placeholder 'Please type into the text field the desired banner' and a note '* PLEASE ENTER THE BANNER TEXT'. Below the text field are 'CANCEL' and 'NEXT' buttons. To the right of the dialog is a list of items with a green '+' button at the top. The items are listed by name (Ascending): Network-Restore, Network-System, Network-Time, and Network-User. Each item has three icons: a rocket (Edit), a thumbs up (Approve), and a trash can (Delete).



Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.4



Red Hat



Red Hat
Ansible Automation
Platform

Section 2.5

Topics Covered:

- Role based access control

Role Based Access Control (RBAC)

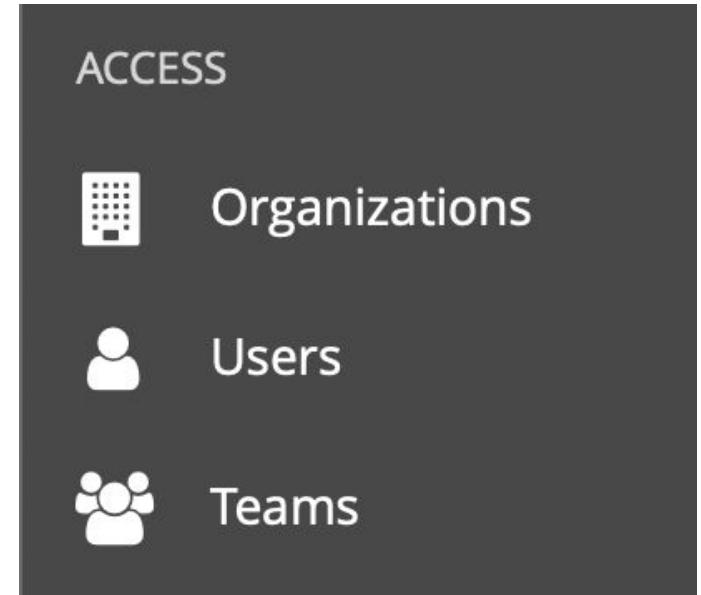
Role-Based Access Controls (RBAC) are built into Ansible Tower and allow administrators to delegate access to inventories, organizations, and more.

These controls allow Ansible Tower to help you increase security and streamline management of your Ansible automation.



User Management

- An **organization** is a logical collection of users, teams, projects, inventories and more. All entities belong to an organization.
- A **user** is an account to access Ansible Tower and its services given the permissions granted to it.
- **Teams** provide a means to implement role-based access control schemes and delegate responsibilities across organizations.



Viewing Organizations

Clicking on the **Organizations** button
will open up the Organizations window



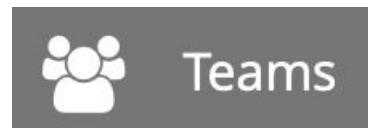
The screenshot shows the Red Hat Satellite 6 interface. On the left, there is a dark sidebar with various navigation options: Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories, Inventory Scripts), Access (Organizations, Users, Teams), and Administration. The "Organizations" option is highlighted with a light blue background. The main content area is titled "ORGANIZATIONS" and contains three organization cards:

- Default**: Contains 0 users, 1 inventory, and 1 job template under the "USERS" section; 0 teams, 1 project, and 0 admins under the "TEAMS" section.
- REDHAT COMPUTE ORGANIZATION**: Contains 0 users, 0 inventories, and 0 job templates under the "USERS" section; 2 teams, 0 projects, and 0 admins under the "TEAMS" section.
- REDHAT NETWORK ORGANIZATION**: Contains 2 users, 1 inventory, and 6 job templates under the "USERS" section; 2 teams, 1 project, and 1 admin under the "TEAMS" section.

At the bottom right of the main content area, it says "ITEMS 1 - 3". The top right corner of the interface shows the user "admin" and various system icons.

Viewing Teams

Clicking on the **Teams** button
will open up the Teams window



Teams

in the left menu

The screenshot shows the Ansible Tower web interface. On the left, there is a dark sidebar with various navigation options: Views (Dashboard, Jobs, Schedules, My View), Resources (Templates, Credentials, Projects, Inventories), ACCESS (Organizations, Users), and ADMINISTRATION (Teams). The 'Teams' option is highlighted with a grey background. The main content area has a header 'TEAMS' with a count of 4. Below the header is a search bar and a 'KEY' button. A table lists four teams: 'Compute T1' and 'Compute T2' belong to 'REDHAT COMPUTE ORGANIZATION', while 'Netadmin' and 'Netops' belong to 'REDHAT NETWORK ORGANIZATION'. Each team entry has edit and delete icons in the 'ACTIONS' column. At the bottom right of the table, it says 'ITEMS 1 - 4'.

NAME	ORGANIZATION	ACTIONS
Compute T1	REDHAT COMPUTE ORGANIZATION	
Compute T2	REDHAT COMPUTE ORGANIZATION	
Netadmin	REDHAT NETWORK ORGANIZATION	
Netops	REDHAT NETWORK ORGANIZATION	

Viewing Users

Clicking on the **Users** button
will open up the Users window



in the left menu

The screenshot shows the Ansible Tower web interface. On the left, there is a dark sidebar with various navigation options: Views, Dashboard, Jobs, Schedules, My View, Templates, Credentials, Projects, Inventories, Inventory Scripts, Organizations, Users (which is highlighted in grey), and Teams. The main content area has a header with "TOWER" and "admin". Below the header is a "USERS" section with a search bar, a "KEY" button, and a green "+" button. A table lists eight users: admin, bbelcher, gbelcher, lbelcher, libelcher, network-admin, network-operator, and tbelcher. Each user row includes columns for Username, First Name, Last Name, and Actions (edit and delete icons). At the bottom of the table, it says "ITEMS 1 - 8".

USERNAME	FIRST NAME	LAST NAME	ACTIONS
admin			
bbelcher	Bob	Belcher	
gbelcher	Gene	Belcher	
lbelcher	Louise	Belcher	
libelcher	Linda	Belcher	
network-admin	Larry	Niven	
network-operator	Issac	Assimov	
tbelcher	Tina	Belcher	



Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.5



Red Hat



Red Hat
Ansible Automation
Platform

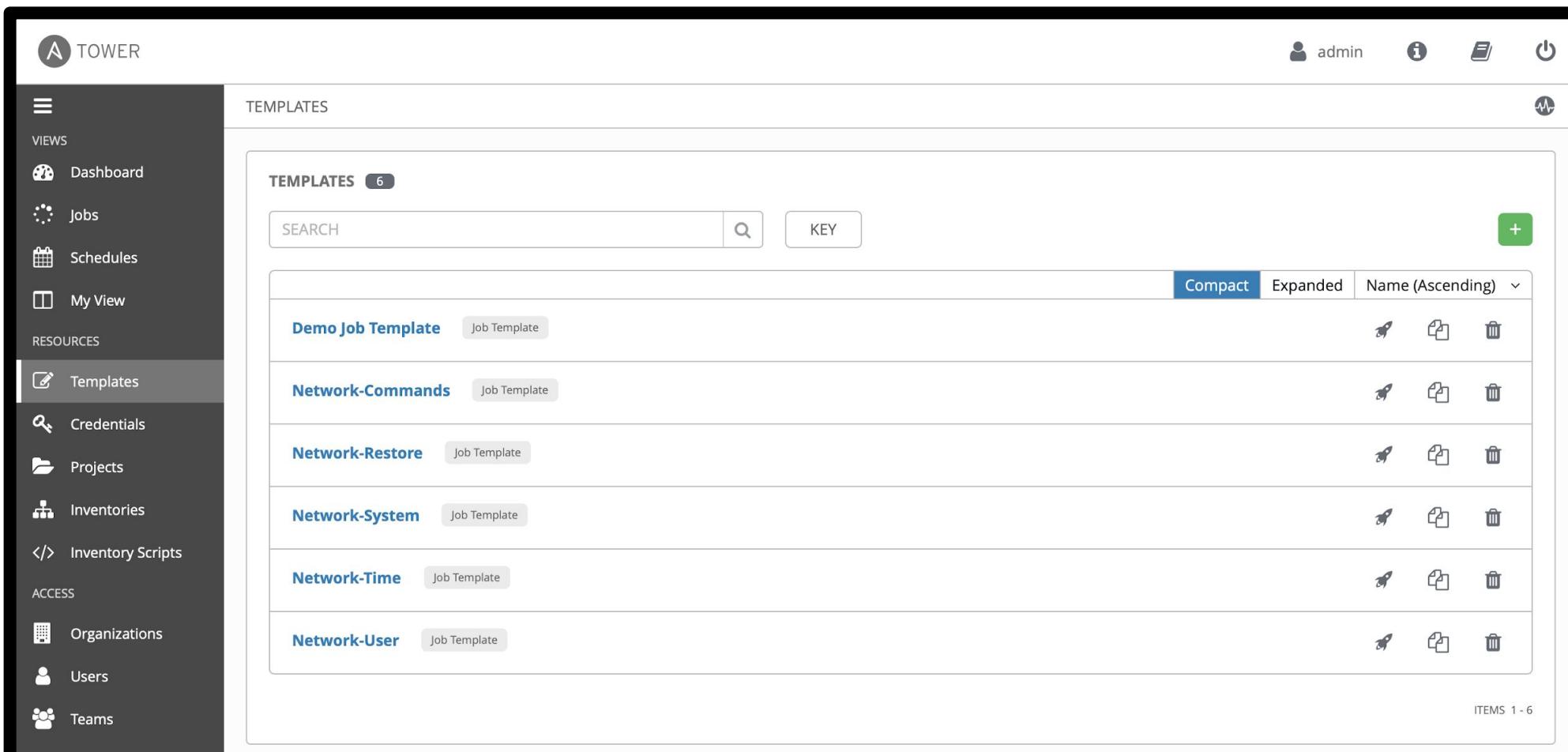
Section 2.6

Topics Covered:

- Workflows

Workflows

Workflows can be found alongside Job Templates by clicking the **Templates**  button under the *RESOURCES* section on the left menu.



The screenshot shows the Ansible Tower web interface. The top navigation bar includes the TOWER logo, user info (admin), and various icons. On the far left is a dark sidebar with a navigation menu:

- VIEWS: Dashboard, Jobs, Schedules, My View
- RESOURCES: **Templates** (selected), Credentials, Projects, Inventories, Inventory Scripts
- ACCESS: Organizations, Users, Teams
- ADMINISTRATION

The main content area is titled "TEMPLATES" and shows a list of six items:

NAME	TYPED AS	OPTIONS
Demo Job Template	Job Template	
Network-Commands	Job Template	
Network-Restore	Job Template	
Network-System	Job Template	
Network-Time	Job Template	
Network-User	Job Template	

At the bottom right of the main content area, it says "ITEMS 1 - 6".

Adding a new Workflow Template

To add a new **Workflow** click on the green + button

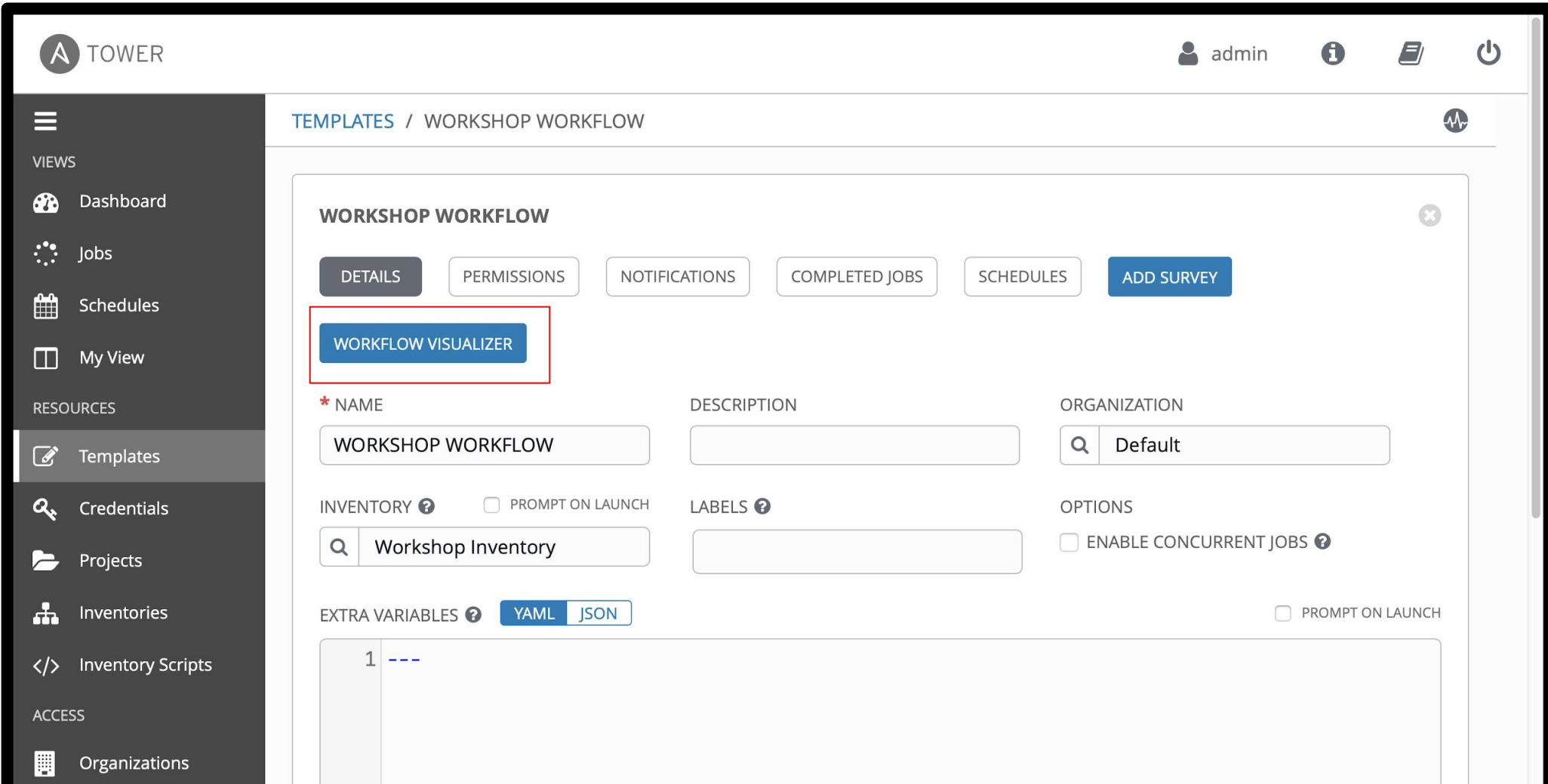


This time select the **Workflow Template**

A screenshot of the Ansible Tower web interface. The left sidebar shows navigation options like Dashboard, Jobs, Schedules, My View, Templates (which is selected), Credentials, Projects, Inventories, Inventory Scripts, Organizations, and Users. The main content area is titled 'TEMPLATES' and shows a list of templates. Each template entry includes a name, a 'Job Template' badge, a preview icon, and three action icons (rocket, copy, delete). A red box highlights the 'Workflow Template' option in the dropdown menu that appears when clicking the green '+' button in the top right corner of the template list. The dropdown also lists 'Job Template'. The interface has a dark mode theme with light-colored cards for each template.

Creating the Workflow

Fill out the required parameters and click **SAVE**. As soon as the Workflow Template is saved the WORKFLOW VISUALIZER will open.



The screenshot shows the Tower interface for creating a new workflow template. The left sidebar has 'Templates' selected. The main area shows the 'WORKSHOP WORKFLOW' template details. The 'WORKFLOW VISUALIZER' button is highlighted with a red box.

WORKSHOP WORKFLOW

DETAILS **PERMISSIONS** **NOTIFICATIONS** **COMPLETED JOBS** **SCHEDULES** **ADD SURVEY**

WORKFLOW VISUALIZER

*** NAME**: WORKSHOP WORKFLOW

DESCRIPTION: (empty)

ORGANIZATION: Default

INVENTORY: Workshop Inventory

LABELS: (empty)

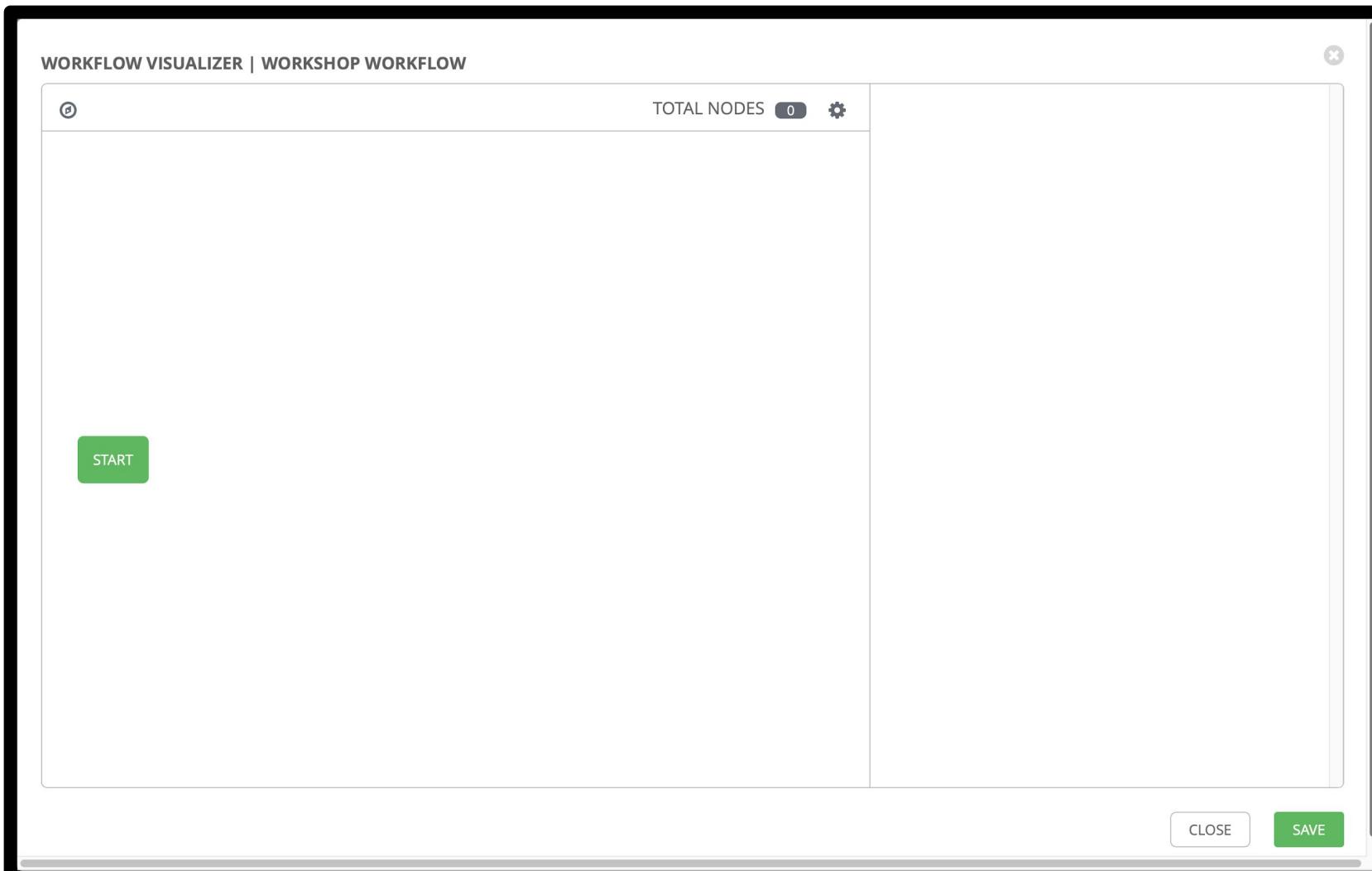
OPTIONS: ENABLE CONCURRENT JOBS

EXTRA VARIABLES: (empty)

YAML **JSON**

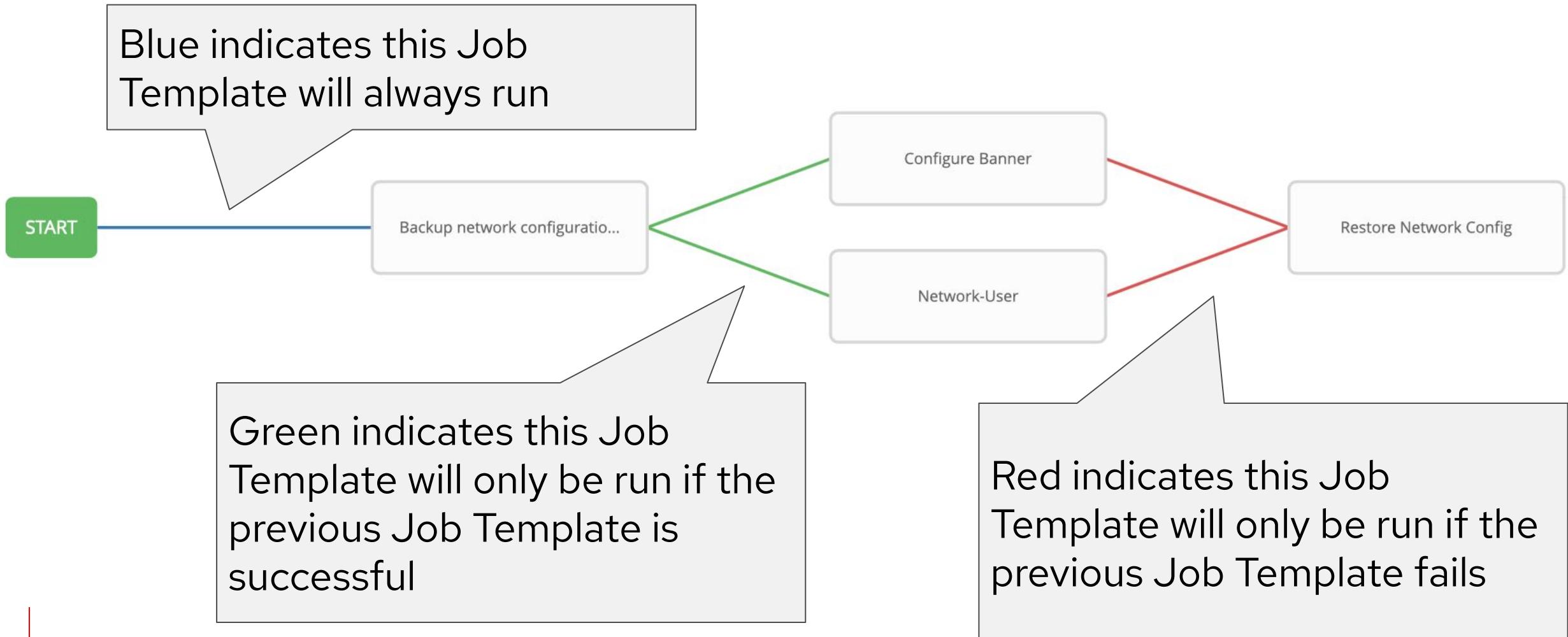
Workflow Visualizer

The workflow visualizer will start as a blank canvas.



Visualizing a Workflow

Workflows can branch out, or converge in.





Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.6



Red Hat



Red Hat
Ansible Automation
Platform

Section 2.7

Topics Covered:

- Wrap-up



Red Hat

Ansible Automation Platform

Demo Time

Exercise 2.7



Red Hat

Training at Red Hat

Customer return on investment from training

365% 3-year ROI

—

IDC conducted a study to explore how Red Hat® training courses impacted the skills, performance, and productivity levels of customers. They found that training for impacted IT professionals and developers consistently increases both individual capability and the ultimate business value of the supported technology.

Other key findings include:

 **44%**
higher DevOps team productivity

 **59%**
faster to deploy new IT resources

 **34%**
more efficient IT infrastructure teams

 **76%**
faster to full productivity, new hires already trained

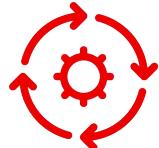
Improve productivity with training in Ansible automation

Scale people, processes, and infrastructure



Red Hat Ansible Automation Platform

A powerful foundation to build and operate automation across organizations. Prepare your teams with the right skills to make the most out of new technology investments.

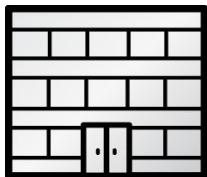


59% faster
deployment of new IT resources

"Red Hat Training shows our DevOps team how to automate a repeatable task. They can write one playbook to execute a set of tasks that would have taken hours or days of time."

"With Red Hat Training it doesn't matter which engineer is engaged on a project. They are all using Ansible for automating tasks, allowing them collectively to be **five times as productive** ... This was not possible previously. As a result, they've definitely picked up the pace of productivity."

WAYS TO TRAIN



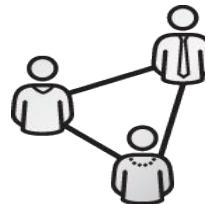
Onsite Training

Private On-site training and exams delivered at your location or at one of our training centers



Classroom Training

Training and test in a professional classroom environment led by Red Hat Certified Instructors



Virtual Training

Live instructor-led online training with the same high-quality, hands-on labs you'd find in our classrooms



Online Learning

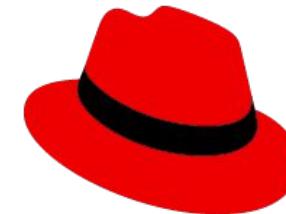
90 days of access to course content and up to 80 hours of hands on labs – all available online, at your pace, and your schedule.

RED HAT LEARNING SUBSCRIPTION

A prescriptive, reliable, learning solution for rapid skills transformation on Red Hat technologies

Simple, flexible, on-demand training

- 24x7 access globally, available offline
- Self-paced, unlimited access to Red Hat courses
- Access to content currently in development
- Updated content pushed as early releases
- Content spanning the entire Red Hat product portfolio
- Early access to completed chapters of courses



**Red Hat
Learning
Subscription**

Red Hat Learning Subscription Evolution

Introducing a Premium subscription tier



STANDARD

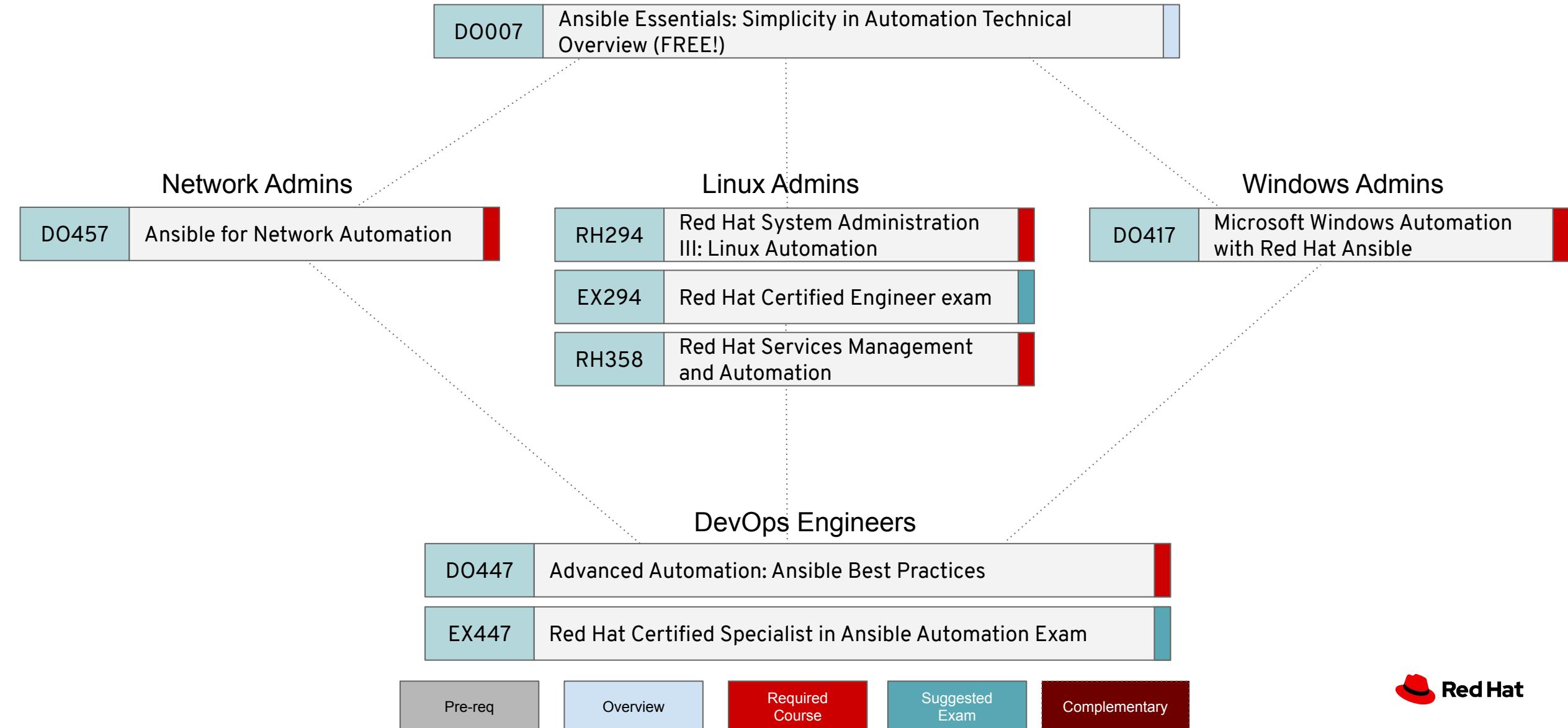


MODULARIZED VIRTUAL
TRAINING



PREMIUM

Ansible Curriculum



Thank you



linkedin.com/company/red-hat



youtube.com/AnsibleAutomation



facebook.com/ansibleautomation



twitter.com/ansible



github.com/ansible